

06-23-00

A

06/21/00
JC860 U.S. PTO

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventorship..... Fisher et al.
Applicant..... Microsoft Corporation
Attorney's Docket No. MS1-558US
Title: Spreadsheet Fields In Text

JC844 U.S. PTO
09/599808
06/21/00

TRANSMITTAL LETTER AND CERTIFICATE OF MAILING

To: Commissioner of Patents and Trademarks,
Washington, D.C. 20231

From: Lewis C. Lee (Tel. 509-324-9256; Fax 509-323-8979)
Lee & Hayes, PLLC
421 W. Riverside Avenue, Suite 500
Spokane, WA 99201

The following enumerated items accompany this transmittal letter and are being submitted for the matter identified in the above caption.

1. Specification—title page, plus 64 pages, including claims 1-80 and Abstract
2. Transmittal letter including Certificate of Express Mailing
3. 10 Sheets Formal Drawings (Figs. 1-10)
4. Return Post Card

Large Entity Status [x]

Small Entity Status []

Date: June 21, 2000

By: L. C. Lee
Lewis C. Lee
Reg. No. 34,656

CERTIFICATE OF MAILING

I hereby certify that the items listed above as enclosed are being deposited with the U.S. Postal Service as either first class mail, or Express Mail if the blank for Express Mail No. is completed below, in an envelope addressed to The Commissioner of Patents and Trademarks, Washington, D.C. 20231, on the below-indicated date. Any Express Mail No. has also been marked on the listed items.

Express Mail No. (if applicable)

EL624352096

Date: June 21, 2000

By: Helen M. Hare
Helen M. Hare

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Spreadsheet Fields In Text

Inventor(s):

Oliver Fisher

Chris Franklin

Alexander Gounares

Matthew Kotler

Matthew Morgan

ATTORNEY'S DOCKET NO. MS1-558US

001290-8086560

TECHNICAL FIELD

This invention relates to computer programs, and particularly, to word processing and spreadsheet programs. More particularly, this invention pertains to an architecture and user interface for integrating individual spreadsheet-like cells into word processing text.

BACKGROUND

Word processing and spreadsheet programs are two well-known and widely used software applications. Word processing programs permit users to draft letters, write books, and create other word-centric documents on a computer. Word processing programs are typically designed with the author in mind by offering tools and user interfaces that make writing easier, such as edit functions (e.g., cut, copy, paste, find, replace, etc.), spell and grammar checking, document formatting, and the like. Examples of word processing programs include "Word" from Microsoft Corporation and "WordPerfect" from Corel Corporation.

Spreadsheet programs enable users to create financial records, accounting spreadsheets, budgets, and other number-centric documents on a computer. Spreadsheet programs are developed with the accountant in mind, focusing on tools and user interfaces that simplify data entry and data manipulation. Spreadsheets typically offer such functionality as in-cell formulas, automatic recalculation as data changes, multi-sheet referencing, cell formatting according to data type (e.g., dates, currency, percentages, etc.), and the like. One example of a spreadsheet program is the "Excel" application from Microsoft Corporation.

In the beginning, spreadsheets and word processing texts were entirely separate from one another. There was essentially no integration. Computer users

1 who wanted to create primarily word-based documents would select a word
2 processing program, while users who wished to produce number-oriented
3 documents turned to spreadsheet programs. Over time, however, word processing
4 users expressed interest in incorporating numbers and a spreadsheet "look" into an
5 otherwise word-dominated document.

6 To accommodate such crossover situations, word processing programs
7 evolved to offer tables, a visual structure that could be used to hold and organize
8 numbers and other types of data. Tables arrange data in columns and rows,
9 thereby emulating the spreadsheet "look". Word processing users can insert a
10 table, modify its layout, and change cell formats to achieve a specific visual
11 appearance to their data. Some tables even support rudimentary functions, such as
12 adding a set of contiguous cells.

13 While visually similar to spreadsheets, word processing tables do not
14 support full spreadsheet functionality. For instance, a user is not able to insert
15 functions into a table that automatically update anytime data is modified in the
16 cells.

17 More recently, object-oriented programming and OLE technologies have
18 been used to provide a richer integration experience. With OLE, word processing
19 users who want greater functionality can embed spreadsheet objects into their
20 word processing documents, instead of tables. Essentially, this is akin to
21 embedding an "Excel" spreadsheet (or other spreadsheet program) into a
22 document running on the "Word" program (or other word processing program).
23 The embedded object carries sufficient functionality to allow the user to enter
24 formulas, format cells, recalculate functions, and do all of the things he/she would
25 normally be able to do on a spreadsheet program.

1 Though the embedded spreadsheet visually resembles a table and provides
2 the desired spreadsheet functionality, it logistically remains a separate program
3 that must be invoked by the user. OLE requires that both types of application
4 programs—a word processor and a spreadsheet—be installed on the computer.
5 When the user wants to update the embedded spreadsheet, the user invokes the
6 spreadsheet object by moving a mouse pointer to anywhere on the embedded
7 object and double clicking the left mouse button (or via some other actuation
8 mechanism). In response, an instance of the spreadsheet program is executed and
9 the spreadsheet changes appearance from a “table look” to a reduced size
10 spreadsheet program with numbered rows and lettered columns and program
11 specific menus. In this state, the user can change functions, modify data, reformat
12 the spreadsheet, and perform other spreadsheet tasks. When the user is finished,
13 the user returns focus to the word processing document by moving the mouse
14 pointer outside the spreadsheet object and single clicking the left mouse button.

15 While the OLE approach offers the full spreadsheet functionality within a
16 word processing document, the process is somewhat sophisticated and typically
17 performed by experienced users who are familiar with both spreadsheets and word
18 processing programs. For novice or less experienced users, it may be confusing to
19 see a table and not appreciate the difference between a word processing table and a
20 full-functioning embedded spreadsheet object. From the user standpoint, different
21 operations are used depending upon whether the visible structure is a table or a
22 spreadsheet. Furthermore, common services such as text formatting, spell
23 checking, and the like do not “tunnel” into the embedded OLE objects and thus,
24 the user is forced to run such services for both the document and the embedded
25 spreadsheet.

1 Thus, even though the final appearance may be visually similar, word
2 processing tables and spreadsheets provide two completely separate mechanisms
3 for displaying information.

4 A separate, but somewhat related problem, is that spreadsheet functionality
5 has been traditionally limited to a grid. There has been no integration of
6 spreadsheet capabilities into word processing text. When writing a document
7 containing both text and a spreadsheet, all data and formulas are typically
8 contained within the spreadsheet of the document. The text then references the
9 values and formula results in the spreadsheet through sentences like, "The total
10 cost of the project is identified in Table 1".

11 If the user actually wants to place a value or formula result in the text, the
12 user manually enters the values directly into the text. For example, the user might
13 write, "The total cost of the project is \$2,300, as identified in Table 1."
14 Unfortunately, the value (e.g., \$2,300) is static text. If the user subsequently
15 changes a value in the spreadsheet, thereby altering the total cost, the sentence
16 remains unchanged. The user must remember the sentence and update it
17 manually.

18 Some word processing programs (e.g., Microsoft Word) allow insertion of
19 "fields" into text. Using fields, a user may enter dates, simple equations, and the
20 like. However, these fields are independent structures that do not recalculate
21 automatically as one would expect for rudimentary spreadsheet functionality.
22 Moreover, these fields tend to be difficult to edit.

23 Accordingly, there is a need to better integrate spreadsheet functionality
24 into text.
25

With the rapidly growing popularity of the Internet, many documents delivered to and rendered on computers are written in markup languages, such as HTML (hypertext markup language). Markup languages can allow authors to easily construct a desired visual layout of the document. Some HTML documents provide tables that look and function as if they were integrated with the surrounding text. For instance, financial Websites commonly offer informative discussions on retirement planning, college savings, or buying a house and include with those discussions one or more tables that invite the user to fill in their personal financial information and goals. When the user finishes entering the data fields, the document appears to make on-the-fly calculations and present the results together with the discussions.

Despite the appearance of in-document calculations, the HTML document is nothing more than an electronic form that receives data entered by the user. When the user completes entry, the HTML document is submitted to a Web server that extracts the user data and makes the appropriate financial calculations. The server places the results in another HTML document and serves the document back to the user's computer. The submit and reply occur very quickly, so the user may be unaware that the HTML document holding the results is different than the HTML document into which he/she initially entered data. In any event, the traditional separation between spreadsheets and text/tables has persisted into the Web-based era.

SUMMARY

A system architecture integrates spreadsheet functionality into text and tables. The architecture allows insertion of discrete individual fields, referred to as "free floating fields", inline with normal textual sentences. In an HTML document, for example, the free floating fields are HTML elements constructed along with text elements and rendered together as an integrated document. Once rendered, the free floating fields present contents that resemble normal text.

The free floating fields offer spreadsheet functionality, including the ability to handle complex formulas, reference values in a separate free floating field or table, and automatically recalculate the formulas when a value changes. The values and formula results can also be formatted (e.g., numbers, date, times, currency, etc.), like a spreadsheet, while remaining part of the normal text. The results of formulas and data values can also be formatted like normal text, using familiar text formatting tools.

Underlying the user interface, one implementation of the architecture separates data handling functions from presentation functions. The architecture includes a user interface manager to manage how the free floating fields appear in a document (e.g., selection, cut, copy, paste, etc.) and to facilitate user entry of formulas and values into the free floating fields. The architecture also has a spreadsheet functionality manager to manage the spreadsheet functions for the free floating fields, such as recalculation, formula handling, referencing, and the like.

The bifurcated architecture supports cross-referencing in which one free floating field references text or data in another free floating field, even though the fields are separate from one another. As part of the cross-referencing, the architecture allows a user to reference other fields using a reference edit operation

1 (e.g., move pointer to the field and click to capture content in the field). The
2 architecture further accommodates automatic universal recalculation throughout
3 all fields in the document. Thus, when a user modifies the contents of one field,
4 the architecture automatically recalculates any formulas in any free floating fields
5 affected by the modification.

6 The architecture also permits a user to reference textual content via free
7 floating fields. When entering a formula in a first free floating field, the user can
8 select a range of normal text. A second free floating field is automatically created
9 around the range of text and a reference to that second free floating field is
10 automatically inserted into the formula in the first free floating field. Upon
11 confirmation, the formula is calculated to insert the referenced text into the
12 sentence. As the text in the second free floating field is updated, the formula in
13 the first field is automatically recalculated.

14 Many other architectural features and UI features are described.

15 16 **BRIEF DESCRIPTION OF THE DRAWINGS**

17 Fig. 1 is a block diagram of an exemplary architecture for integrating
18 spreadsheet functionality into word processing text and tables.

19 Fig. 2 illustrates a screen display of a rendered document having two free
20 floating fields and a single table that are capable of spreadsheet functionality. In
21 Fig. 2, the free floating fields exhibit a normal "text look" and the table exhibits a
22 "table look" during a non-editing mode.

23 Fig. 3 illustrates a screen display of the rendered document, where the free
24 floating fields exhibit a "cell look" and the table exhibits a "spreadsheet look"
25 during an editing mode.

1 Fig. 4 illustrates a screen display of another rendered document, where a
2 free floating field references text in the document.

3 Fig. 5 illustrates a screen display of yet another rendered document having
4 multiple tables. In particular, Fig. 5 shows nested tables, where one table is
5 inserted into a cell of another table, and the ability to reference from one table to
6 the other table.

7 Fig. 6 illustrates a screen display of another rendered document having
8 multiple tables and a free floating field that appear in an edit mode.

9 Fig. 7 is a block diagram of an exemplary computer that implements the
10 architectures of Figs. 1 and 4.

11 Fig. 8 is a flow diagram of a process implemented by the architecture of
12 Fig. 1.

13 Fig. 9 is a diagrammatic illustration of how a user interface table in a
14 rendered document and underlying functional components in the architecture work
15 together during a recalculation operation.

16 Fig. 10 is a diagrammatic illustration of how multiple free floating fields
17 and UI tables and underlying functional components in the architecture work
18 together during a cross-table reference edit operation and a concurrent
19 recalculation operation.

20 21 **DETAILED DESCRIPTION**

22 This disclosure describes an architecture that integrates spreadsheet
23 functionality into word processing text. The architecture allows insertion of
24 discrete individual fields, referred to as "free floating fields", inline with normal
25 textual sentences. The free floating fields offer spreadsheet functionality,

1 including the ability to handle complex formulas, reference values in a separate
2 free floating field or table, and automatically recalculate the formulas when a
3 referenced value changes. The values and formula results can also be
4 independently formatted (e.g., numbers, date, times, currency, etc.), like
5 spreadsheet cells, while remaining part of the normal text of a document. The
6 results of formulas and data values may also be formatted like normal text, using
7 familiar text formatting tools.

8 The architecture also supports integration of spreadsheet functionality into
9 word processing tables. The tables resemble a spreadsheet when being edited, and
10 a table when not being edited. The tables enjoy the benefits of word processing
11 tables (e.g., size and shape formatting, spell and grammar checking, etc.) as well
12 as the benefits of spreadsheets (e.g., complex formulas, sorting, recalculation,
13 etc.). The architecture also allows cross referencing among tables and free
14 floating fields such that, when the content of one table cell or free floating field is
15 modified, the architecture automatically recalculates any formulas in all tables and
16 free floating fields that are affected by the modification. With this architecture,
17 spreadsheet functionality is no longer constrained to grids, but is available in
18 customary text and across table boundaries.

19 For discussion purposes, the architecture is described in the context of
20 creating tables, text, and free floating fields in a document written in a markup
21 language (e.g., HTML). In this manner, the user is afforded the rich HTML
22 formatting options of both text and tables, including table layout changes (e.g.,
23 merging and splitting cells), as well as the calculation and formatting features
24 specific to data that are traditionally associated only with a separate spreadsheet
25

1 application. However, it is noted that the architecture may be useful in other
2 document types that are not rooted in a markup language.

3 4 Architecture

5 Fig. 1 shows an exemplary architecture 100 that integrates spreadsheet
6 functionality into word processing text. The architecture 100 may be implemented
7 on a standalone computer, a network server computer, a network client computer,
8 or distributed at both the server and client. The architecture 100 includes a
9 document renderer 102, one or more sets of table objects 104, one or more sets of
10 spreadsheet objects 106, one or more sets of free floating field objects 107, a
11 spreadsheet editor 108, a workbook 110, a spreadsheet engine 112, and one or
12 more non-core worksheet functions 114(1)-114(W) that may be optionally used by
13 the spreadsheet engine 112.

14 The architecture 100 separates data handling functions from presentation
15 functions of the integrated spreadsheet functionality. In this manner, the
16 architecture may be characterized as a cooperation of two system managers: a user
17 interface (UI) manager 116 and a spreadsheet functionality manager 118. The UI
18 manager 116 manages how the free floating fields and tables appear in a document
19 and facilitates such tasks as table resizing, selection, cut, copy, paste, split, merge,
20 table formatting and so on. The UI manager 116 includes the table object 104, the
21 spreadsheet objects 106, and the spreadsheet editor 108.

22 The spreadsheet functionality manager 118 manages the spreadsheet
23 functions for the table and free floating fields, such as recalculation, formula
24 handling, sorting, referencing, and the like. The spreadsheet functionality
25 manager 118 includes the workbook 110, the spreadsheet engine 112 and

1 worksheet functions 114. With the bifurcated architecture, the spreadsheet
2 functionality manager 118 is not concerned with the table layout or other visual
3 features, and the UI manager 116 is not concerned with data management,
4 formulas, and recalculation processes.

5 The bifurcated architecture 100 is advantageous in that it supports cross-
6 referencing among the free floating fields and tables. It also allows reference
7 editing during formula entry to allow convenient selection of other fields or cells
8 and capturing of their contents as references used in the formula. The architecture
9 further facilitates automatic universal recalculation of any formulas throughout all
10 free floating fields and tables in the document that are affected by user
11 modification of a single reference. These goals could also be achieved using
12 different, non-bifurcated architectures.

13 A document 120 is constructed and rendered on the document renderer 102.
14 The document 120 combines a text-based body 122, one or more tables 124(1) and
15 124(2), and one or more free floating fields (FFF) 126(1) and 126(2). Each free
16 floating fields 126 is akin to a single cell of a table that may be inserted anywhere
17 in the document, including in the middle of a text-based body.

18 For discussion purposes, the document 120 is written in a markup language,
19 such as XML (extensible markup language). XML documents can be transformed
20 using XSL (extensible stylesheet language) and rendered directly as HTML
21 (hypertext markup language). In this case, the renderer 102 may be implemented
22 as a browser or other application that handles and renders HTML documents. The
23 tables 124 are thus rendered as HTML tables. Examples of various documents
24 that contain integrated spreadsheet functionality into text are illustrated in Figs. 2-
25 6.

1 With continuing reference to Fig. 1, the table and spreadsheet objects 104
2 and 106 provide editing functionality for the table 124, including such functions as
3 table resizing, selection, cut, copy, paste, split, merge, table formatting, and a host
4 of other rich spreadsheet events. The free floating field objects 107 provide
5 similar editing functionality for the free floating fields 126, with the exception that
6 certain functions do not make sense in the context of a free floating field. For
7 example, it does not make sense to "split" a free floating field as one would split a
8 table cell.

9 The spreadsheet engine 112 provides the spreadsheet functionality for the
10 table 124 and free floating fields 126, including such functions as formula
11 creation, reference editing, recalculation, and the like. Architecturally, the
12 spreadsheet components are separate from the word processing table and text,
13 although the spreadsheet relies on the table and free floating fields and the table
14 and free floating fields provide special notifications and events to help the
15 spreadsheet. This allows either component to add additional functionality without
16 directly affecting the other component.

17 The spreadsheet engine 112 has a grid object for each table and free
18 floating field in the document 120, as represented by grid objects 130(1) and
19 130(2). The grid objects 130 receives events indicative of user activity in the table
20 124 and free floating fields 126 and coordinates actions among various objects.
21 The workbook 110 initially creates the tables 124 and free floating fields 126, and
22 subsequently tracks all grid objects 130 to resolve any cross-table or free floating
23 field referencing. Upon creation, a grid object 130 registers with the workbook
24 110 so that it can participate when tables and free floating fields are updated. The
25 grid object 130 keeps an interface to the spreadsheet behavior 106 and the free

floating field behavior 107 to fetch values from the HTML tree maintained at the renderer 102.

Each grid object 130 maintains two structures: a format table 132 and a cell table 134. The format table 132 holds information about the data format of each cell in the table 124 or associated free floating field 126. For instance, the cell or field may contain dates, numbers, dollar amounts, percentages, and so forth. In the example shown in Fig. 3, the format table 132 would contain information that cells A1-A6, B1-B6, and C1, are text and cells C2-C6 are formatted as currency in U.S. dollars. Also in Fig. 3, the format table 132(2) for free floating field 126(1) contains information that the field is formatted as currency in U.S. dollars.

The cell table 134 stores the actual data for each cell in the table 124 or an associated free floating field 126, such as text, values, and formulas. A cell table 134 stores pointers to one or more cells that physically store the data. A cell table associated with a table, such as cell table 134(1), contains multiple cells 136(1)-136(C), one for each cell in the table. A cell table associated with a free floating field, such as field 134(2), contains only one cell 136 because the free floating field 126(1) can be thought of as a table with only one cell.

Each cell 136 is an object containing the parsed value of the cell and a reference to complete information about the cell. If the cell contains text or numeric data (e.g., cells A1-A6, B1-B5, and C1-C5 in Fig. 3), it is stored directly in the object. Formulas, such as the summation formula in cell C6 of Fig. 3, are stored as pointers to the appropriate formula object maintained by the formula manager 140 (discussed below).

The spreadsheet engine 112 includes a formula manager 140 to handle all formulas and parsing duties for formulas, data values, and references (e.g.,

1 D4:E23). The workbook 110 serves as the linkage between the formula manager
2 140 and the registered grids 130. The formula manager 140 maintains a
3 recalculation engine 142 that performs recalculation of all formulas in response to
4 event changes in the table or free floating fields. In one implementation, the
5 recalculation engine 142 maintains the formulas for a document in a bi-directional
6 linked list, sometimes referred to as the "formula chain". Following a
7 recalculation event (e.g., user entry of a new data value or new formula), the
8 recalculation engine 142 traverses the list, evaluating formulas that may be
9 affected by the event.

10 If the current formula depends on other formulas that have not yet been
11 evaluated, the current formula is moved to the end of the list. After one
12 recalculation pass, the formula list is organized in natural order and will not need
13 to be reordered during subsequent recalculations unless new formulas are added.
14 If recalculation comes to a formula that has already been bumped to the end of the
15 list and discovers that this formula still relies on not-yet-calculated dependencies,
16 the formula contains a circular reference. In this case, the recalculation engine
17 returns a circular error.

18 The formula manager 140 also has a parser 144 that parses the formulas. In
19 one implementation, the parser 144 is a recursive descent parser that extracts
20 tokens from a stream and appends them to an array of character-size operation
21 types and a parallel array of variant operands. When done, the parser 144 creates
22 a new formula object 146 and gives it the two arrays of parsed information. The
23 formula manager 140 therefore maintains one or more formula objects 146(1)-
24 146(B) that contain formula information, including the parsed formula expression
25

1 returned by the parser 144, the current result, the type of formula, and the current
2 formula state.

3 In one implementation, there are three types of formulas: normal, semi-
4 calculation, and non-calculation. The normal formula is reevaluated only when its
5 dependencies change. The semi-calculation formula is reevaluated every time the
6 recalculation engine 142 performs a recalculation operation. The non-calculation
7 formula is never evaluated at all. Non-calculation formulas are a special formula
8 type for handling nested tables (i.e., a table within a table) and nested free floating
9 fields.

10 Consider the case of an inner table nested inside a cell of an outer table. If
11 the inner table contains a formula that changes to a different value following
12 recalculation, the value of the outer table will also change. Such a dependency is
13 not encoded anywhere, since there is no formula attached to the outer table. In
14 such cases, a non-calculation formula is set to re-fetch the resulting value from the
15 inner calculation. Thus, it participates in the normal dependency management of
16 recalculation and all references to the outer table are updated when appropriate.

17 In one implementation, the formula objects 146 are owned by a COM
18 wrapper (not shown), which is in turn held onto by a cell object 136 in the grid
19 130 where the formula resides. The formula objects 146 are themselves part of the
20 bi-directional linked list of formulas maintained by the recalculation engine 142.
21 The formula objects 146 contain references to the table row and column (or the
22 free floating field) they are stored in and to the cell object 136 in grid 130. The
23 references allow the recalculation engine 142 to travel down the recalculation
24 chain with formulas from several tables and easily determine to which table a
25

1 given formula belongs. Many operations, from formula saving to table deletion,
2 depend on this ability to traverse the chain.

3 The formula manager 140 also parses referenced cell groups. As examples,
4 the formula manager 140 parses "A5" as a cell reference, "D4:E23" as a
5 compound rectangular reference, "\$F\$30" as an absolute reference, "Table5!D5"
6 as a cross-table reference, "Field3" as a whole-table cross-table reference, "A5:D5
7 B3:B6" as an intersection, and "D3,E4" as a union.

8 The non-core worksheet functions 114(1)-114(W) are optional elements.
9 Examples of such functions include analysis functions, statistical functions, and
10 trigonometric functions. The modular architecture 100 makes it flexible to remove
11 unwanted worksheet functions or add new worksheet functions.

12 The spreadsheet objects 106 and free floating field objects 107 are
13 counterparts to their corresponding grid objects 130(1) and 130(2) and are located
14 outside of the spreadsheet engine 112. There is one pair of a spreadsheet object
15 106 and a grid object 130(1) per table 124 and one pair of corresponding free
16 floating field behavior 107 and grid object 130(2) per free floating field 126.

17 The spreadsheet objects 106 and free floating field objects 107 define
18 behaviors that receive events from the document renderer 102, process them, and
19 pass the events onto the corresponding grid objects 130(1) and 130(2). In
20 response to the events, the grid objects 130(1) and 130(2) update the cell data in
21 cell tables 134(1) and 134(2) and/or formatting information in format tables
22 132(1) and 132(2).

23 The spreadsheet behavior 106 has three objects: GridBehavior 150,
24 CellEditing 152, and Spreadsheet 154. The GridBehavior object 150 provides a
25 layer of abstraction between the grid object 130 and individual HTML table cells

1 and allows the grid object 130 to access HTML values and styles. The
2 GridBehavior object 150 wraps the HTML elements in a common interface so that
3 the grid 130 does not need to know the particular structure of the HTML table.
4 Additionally, the GridBehavior object 150 manages table-specific portions of a
5 "reference edit" operation.

6 The CellEditing object 152 and Spreadsheet object 154 interact directly
7 with an HTML tree and the table behavior 104 to provide the grid 130 with events.
8 The Spreadsheet object 154 is responsible for recording undo records for actions
9 affecting the spreadsheet.

10 The CellEditing object 152 manages user-level editing of cells. It processes
11 events related to user edits of in-cell data values and provides some editing user
12 interface (UI) elements, including the formula edit box that permits user edits of
13 formulas. When editing a formula, a floating formula edit box is provided above
14 the cell's location and resized as necessary to accommodate the formula. The
15 localized edit box eliminates a potential UI problem of forcing the user to fit the
16 entire formula into the table cell, which may cause the table to resize strangely as
17 the user clicked into and out of the cell.

18 The CellEditing object 152 also supports the reference edit operation when
19 the formula edit box is presented. As noted above, the reference edit operation
20 allows the user to visually reference cells using a mouse pointer (or other focus
21 mechanism) and in response, inserts a reference to that cell data in the current
22 formula edit box. The formula edit box is described below in more detail. The
23 CellEditing object 152 is only present when a cell is being actively edited.

24 The free floating field behavior 107 has three objects: an FFFBehavior
25 object 160, a CellEditing object 162, and a Spreadsheet object 164. The

The FFFBehavior object 160 takes the place of GridBehavior object 150 and the Table object 104 in the context of free floating fields. Like the Table object 104, the FFFBehavior object 160 monitors the document for changes to the free floating field, although these are much more limited than in the case of a table. Like the GridBehavior object 150, the FFFBehavior object 160 provides an interface for the grid object 130(2) and manages “reference edit” operations for the free floating field.

The spreadsheet editor 108 handles top-level duties such as inserting a table or a free floating field and routing commands to the appropriate table based on the current selection in the document 120. The editor 108 also creates and manages the workbook 110.

The integrated spreadsheet table and free floating field model eliminates the need for the user to choose the structure of data within a document prior to creating that document. Historically, if the user needed more control over the presentation of the tabular data, the user tended to select a word processing application. On the other hand, if the user required computations over the data, the user typically chose a spreadsheet application. The integrated architecture allows the user to combine several different types of data within one document.

1 Additionally, by integrating spreadsheet functionality inside a table and
2 inline text fields, the user can build the document as they want. In spreadsheet
3 applications, the user is restricted to the grid layout for all document content. In
4 the integrated architecture, users can create a rich document that contains text
5 integrated with multiple tables and multiple free floating fields, each with data that
6 can be formatted as values and used in calculations throughout different tables and
7 fields.

8 9 **Examples of Documents with Integrated Spreadsheet Functionality**

10 Fig. 2 shows an example document 120 that has text body 122. In the Fig.
11 2 example, the document 120 is a letter written to Mr. Jones describing various
12 home improvement projects and the costs associated with the projects. A single
13 table 124 is situated in the text body. In the depicted non-editing view, the table
14 124 resembles a standard word processing table with three columns 202(1)-202(3)
15 and five rows 204(1)-204(5).

16 Two free floating fields 126(1) and 126(2) are inserted in the text body 122
17 inline with the text. The free floating fields 126 appear as normal text in the
18 document 120. For instance, the second sentence recites, "The total cost of the
19 project is \$12,060". The displayed value "\$12,060" is a result of a formula in the
20 underlying free floating field, which is currently hidden in this view. The formula
21 sums the project costs listed in the table 124.

22 Fig. 3 shows the same document 120 during editing, where the table 124
23 and free floating fields 126(1) and 126(2) are shown as they would look upon
24 selection for editing. Notice that table 124 now "looks" like a spreadsheet more
25 than a traditional table. The table 124 has integrated column headers 302(1),

302(2), and 302(3), as well as integrated row headers 304(1)-304(6). The table 124 has a column expansion control 306 and a row expansion control 308 to permit easy expansion of the table.

It is noted that Fig. 3 is merely an illustration for discussion purposes. Typically, only one of the fields or the table is edited at a time, and hence that field or table exhibits the spreadsheet look (not all three as shown in Fig. 3). Thus, selecting the table for editing invokes the column and row headers 302 and 304. Similarly, selecting a free floating field containing a formula would display the formula edit box. If, on the other hand, a free floating field does not contain a formula it will be selected in the same manner as all surrounding text is selected.

In this example, the user has added a row to the table and is entering a summation formula in cell C6 using a reference edit operation. With a mouse pointer 310, the user references an array of cells C2 through C5 for entry into the formula. Upon confirmation (e.g., releasing the left mouse button), a reference to the cells C2-C5 are inserted into the summation formula in cell C6 and after clicking out of the cell the formula is calculated to add the dollar amounts in column C. The result of \$12,060 is inserted into cell C6.

The free floating fields 126(1) and 126(2) contain formulas that reference values and/or formulas located elsewhere in the document 120. The first free floating field 126(1) contains a formula that sums the dollar amounts in cells C2-C5 of the table 124, or “=SUM(Table1!C2:Table1!C5)”. The second free floating field 126(2) contains a formula that references both the first free floating field 126(1) and a value in the table. Namely, the formula in the second free floating field 126(2) computes the difference between the results of the formula in the first

1 free floating field 126(1) and the cost of the kitchen repair in cell C3, or “=FFF1-
2 Table1!C3”.

3 If a value in cells C2-C5 in the table 124 is modified, the free floating fields
4 126(1) and 126(2) are automatically updated to reflect the modification. For
5 instance, suppose the user modifies the cost of the family room in cell C4 from
6 \$3,500 to \$4,000. As a result of this change, the formula in table cell C6 is
7 automatically recalculated to yield \$12,560, the formula in the first free floating
8 field 126(1) is automatically recalculated to yield \$12,560, and the formula in the
9 second free floating field 126(2) is automatically recalculated to yield \$7,060.

10 Fig. 4 shows another example of a document 400 in which a free floating
11 field 402 is used to reference text, rather than data values. In this case, free
12 floating fields enable a new mechanism for the reference edit operation. As shown
13 in Fig. 3, the typical reference edit operation allows entry of formula references
14 into a spreadsheet by pointing at (i.e., clicking on) a referenced cell or range of
15 cells while editing the formula. The referenced cells are highlighted automatically
16 and the correct reference to them is inserted into the formula. Free floating fields
17 extend the reference edit operation to allow a range of normal text in a document
18 to be referenced in a formula.

19 In contrast to this typical reference edit operation, the free floating field 402
20 in document 400 contains a formula that references a text range 404. Using a
21 mouse pointer 406, the user selects a range of normal text 404 and the text is
22 highlighted. A new free floating field is automatically created around the range of
23 text 404 and a reference to that new free floating field, FFF1, is automatically
24 inserted into the formula being edited within the formula edit box 408. In this
25 example, the edited formula is in free floating field 402; however, in other cases,

1 the formula may be in a table. Upon confirmation (e.g., pressing the return key),
2 the formula is calculated and the referenced text "Formal Bid" is inserted into the
3 sentence as part of the normal context flow.

4 If the referenced text 404 is modified, the free floating field 402 is
5 automatically updated to reflect the modification. For instance, suppose the user
6 changes "Formal" to "Final". As a result, the free floating field 402 is
7 automatically recalculated to hold the words "Final Bid".

8 Fig. 5 shows a document 500 that contains a first or outer table 502 and a
9 second or inner table 504 nested within cell B3 of the outer table 502. The ability
10 to nest tables is one feature of this architecture that conventional spreadsheet
11 programs do not provide. Fig. 5 further illustrates a reference edit operation
12 across table boundaries in which the inner table 504 is being referenced for
13 insertion into a formula in cell C3 of the outer table 502.

14 It is noted that the inner table 504 may be replaced by a free floating field,
15 akin to a nested one cell table. Moreover, the outer table 502 may be replaced by
16 a free floating field. In this manner, one or more free floating fields may be nested
17 within one or more table cells, an entire table may be nested within a free floating
18 field, and one or more free floating fields may be nested within one or more other
19 free floating fields.

20 Fig. 6 shows a document 600 having three tables 602, 604, and 606, and a
21 free floating field 608. Document 600 illustrates a situation where multiple tables
22 and fields can cross-reference one another.

23 The many features of the user interface presented through the various
24 rendered documents are discussed in greater detail below under the section
25 heading "User Interface Features".

Exemplary Computing Environment

Fig. 7 illustrates an example of an independent computing device 700 that can be used to implement the architecture of Fig. 1. The computing device 700 may be implemented in many different ways, including a general-purpose computer (e.g., workstation, server, desktop computer, laptop computer, etc.), a handheld computing device (e.g., PDA, PIM, etc.), a portable communication device (e.g., cellular phone with computing capabilities), or other types of specialized appliances (e.g., set-top box, game console, etc.).

In the illustrated example, computing device 700 includes one or more processors or processing units 702, a system memory 704, and a bus 706 that couples the various system components including the system memory 704 to processors 702. The bus 706 represents one or more types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. The system memory 704 includes read only memory (ROM) 708 and random access memory (RAM) 710. A basic input/output system (BIOS) 712, containing the basic routines that help to transfer information between elements within the computing device 700 is stored in ROM 708.

Computing device 700 further includes a hard drive 714 for reading from and writing to one or more hard disks (not shown). Some computing devices can include a magnetic disk drive 716 for reading from and writing to a removable magnetic disk 718, and an optical disk drive 720 for reading from or writing to a removable optical disk 722 such as a CD ROM or other optical media. The hard drive 714, magnetic disk drive 716, and optical disk drive 720 are connected to the

1 bus 706 by a hard disk drive interface 724, a magnetic disk drive interface 726,
2 and an optical drive interface 728, respectively. Alternatively, the hard drive 714,
3 magnetic disk drive 716, and optical disk drive 720 can be connected to the bus
4 706 by a SCSI interface (not shown). It should be appreciated that other types of
5 computer-readable media, such as magnetic cassettes, flash memory cards, digital
6 video disks, random access memories (RAMs), read only memories (ROMs), and
7 the like, may also or alternatively be used in the exemplary operating environment.

8 A number of program modules may be stored on ROM 708, RAM 710, the
9 hard disk 714, magnetic disk 718, or optical disk 722, including an operating
10 system 730, one or more application programs 732, other program modules 734,
11 and program data 736. As one example, the architecture 100 may be implemented
12 as one or more programs 732 or program modules 734 that are stored in memory
13 and executed by processing unit 702. The drives and their associated computer-
14 readable media provide nonvolatile storage of computer-readable instructions, data
15 structures, program modules and other data for computing device 700.

16 In some computing devices 700, a user might enter commands and
17 information through input devices such as a keyboard 738 and a pointing device
18 740. Other input devices (not shown) may include a microphone, joystick, game
19 pad, satellite dish, scanner, or the like. In some instances, however, a computing
20 device might not have these types of input devices. These and other input devices
21 are connected to the processing unit 702 through an interface 742 (e.g., USB port)
22 that is coupled to the bus 706. In some computing devices 700, a display 744
23 (e.g., monitor, LCD) might also be connected to the bus 706 via an interface, such
24 as a video adapter 746. Some devices, however, do not have these types of display
25

1 devices. Computing devices 700 might further include other peripheral output
2 devices (not shown) such as speakers and printers.

3 Generally, the data processors of computing device 700 are programmed by
4 means of instructions stored at different times in the various computer-readable
5 storage media of the computer. Programs and operating systems are typically
6 distributed, for example, on floppy disks or CD-ROMs. From there, they are
7 installed or loaded into the secondary memory of a computing device 700. At
8 execution, they are loaded at least partially into the computing device's primary
9 electronic memory. The computing devices described herein include these and
10 other various types of computer-readable storage media when such media contain
11 instructions or programs for implementing the steps described below in
12 conjunction with a microprocessor or other data processor. The service system
13 also includes the computing device itself when programmed according to the
14 methods and techniques described below.

15 For purposes of illustration, programs and other executable program
16 components such as the operating system are illustrated herein as discrete blocks,
17 although it is recognized that such programs and components reside at various
18 times in different storage components of the computing device 700, and are
19 executed by the data processor(s) of the computer.

20 It is noted that the computer 700 may be connected to a network via a wire-
21 based or wireless connection to interact with one or more remote computers. In
22 this network context, the computer 700 may be configured to store and execute
23 portions of the architecture 100, while one or more remote computers store and
24 execute other portions of the architecture. For example, the document renderer
25 102 may reside on one computer, while the remaining components reside on a

1 separate computer. As a result, the architecture is distributed, with various
2 components being stored on different computer-readable media.

3 4 **General Operation**

5 Fig. 8 shows a table/spreadsheet process 800 implemented by the
6 architecture 100 of Fig. 1. The process 800 may be embodied in software stored
7 and executed on a computer, such as computing device 700 in Fig. 7.
8 Accordingly, the process 800 may be implemented as computer-executable
9 instructions that, when executed on a processing system such as processor unit
10 702, perform the operations and tasks illustrated as blocks in Fig. 8.

11 At block 802, the architecture 100 creates a corresponding pair of free
12 floating field and grid objects 107 and 130 for a new free floating field presented
13 as part of the document. In one implementation, the FFFBehavior object 160,
14 CellEditing object 162, and Spreadsheet object 164 are initially created for the
15 new free floating field and then the Spreadsheet object 164 creates an associated
16 grid object 130(2) in the spreadsheet engine 112. The grid object 130(2) includes
17 a format table 132(2) and a cell table 134(2). If this is the first spreadsheet, a
18 workbook 110 is also created. The grid 130 and workbook 110 then create other
19 objects, including the formula manager 140 and a cell object 136.

20 At block 804, in response to the user entering data and/or formulas into the
21 free floating field, the architecture receives the user entry and passes it to the
22 spreadsheet engine 112 for processing. More specifically, in the continuing
23 exemplary implementation, the Spreadsheet object 164 receives a notification
24 from the document renderer 102 and passes it along to the grid 130(2) for the new
25 free floating field.

1 At block 806, based on the user input, the architecture determines whether
2 to parse the user-entered information or delay parsing until later. The architecture
3 preferably employs a delay parser that parses fields and table cells when
4 necessary, such as the first time that a formula has been loaded or the first time a
5 value has been edited or referenced. If the field contains a formula, the cell is
6 parsed immediately and added to the recalculation chain. If the field does not
7 contain a formula, it is left unparsed until a formula requires its value.

8 At block 808, assuming that parsing is needed now (i.e., the "no" branch
9 from block 806), the architecture parses the user-entered information and updates
10 the format table 132 and cell table 134 based upon this information. At block 810,
11 the architecture determines whether recalculation is appropriate. Some user input
12 may not require recalculation, such as insertion of a new data value that is not
13 referenced by a formula. If recalculation is not needed, flow continues at block
14 814 to determine whether the fields need to be updated as a result of the user
15 input.

16 At block 812, assuming recalculation is appropriate (i.e., the "yes" branch
17 from block 810), the architecture recalculates the various formulas that may have
18 been affected by the user input. In the ongoing example, the act of putting a value
19 into a cell 136 in the cell table 134 triggers a data-changed notification to
20 registered listeners, which includes the grid objects 130(1) and 130(2). The grid
21 objects identify any changed cells and forward the notification to the formula
22 manager 140, which marks any formulas depending on the changed cells as dirty
23 and in need of recalculation.

24 The grid objects 130 then call the recalculation engine 142, which loops
25 over the recalculation formula chain and recomputes the variously affected

1 formulas (block 812(1)). While evaluating the formulas in the formula chain,
2 unparsed cells that were previously left unparsed may now be parsed (block
3 812(2)).

4 Once the recalculation engine 142 finishes, it returns control to the
5 workbook 110. The workbook 110 calls back to the recalculation engine 142 to
6 learn which formulas changed as a result of the recalculation cycle. The
7 workbook 110 then locates the grid objects 130 that hold the formulas and calls
8 them to save the cells that contain the formulas.

9 At block 814, the architecture 100 determines whether any results have
10 changed following the recalculation. If results have not changed (i.e., the “no”
11 branch from block 814), process flow continues at block 804 for the next user
12 input. Conversely, if the results have changed (i.e., the “yes” branch from block
13 814), the architecture 100 loads an updated free floating field with the modified
14 value and renders the updated free floating field as part of the document (block
15 816).

16 It is noted that the above example assumed that the user entered data or a
17 formula. The user may also change the format of the free floating field. The
18 process 800 handles format changes in essentially the same way, but accounts for
19 those changes in the format table 132(2) rather than the cell table 134(2).
20

21 User Interface Features

22 The architecture 100 supports many user interface (UI) features in the
23 rendered document to convey integration of spreadsheet functionality into word
24 processing tables and text. These UI features are described separately below, with
25 reference to Figs. 2-6. Hand-in-hand with these features are the underlying

1 operations and inner workings of various components in the architecture 100.
2 These aspects will be described in more detail later in this disclosure under the
3 heading "Functionality Features".

4 One of the primary benefits of integrating spreadsheet functionality into
5 text and tables is that the user need no longer think in terms of whether the
6 document should be primarily a spreadsheet document produced by a spreadsheet
7 application (e.g., an ".xls" file from the "Excel" program) or primarily a word
8 processing document produced by a word processing application (e.g., a ".doc"
9 file from the "Word" program). Instead, the user creates an HTML document (or
10 other markup-based document, such as an XML document) that can have both text
11 and spreadsheet/table components. By integrating spreadsheet functionality into
12 text, the user can build the document with automatically updateable fields without
13 being restricted to the grid layout.

14 15 Integrated Headers

16 The table 124 toggles between a "table look" (Fig. 2) and a "spreadsheet
17 look" (Fig. 3) depending upon whether the user is editing the table. The
18 spreadsheet look may be invoked in a number of ways, including by actively
19 editing a cell, by hovering a pointer over the table, or by some other activity. As
20 illustrated in Fig. 3, the spreadsheet look includes column headers 302(1)-302(3)
21 and row headers 304(1)-304(6) that integrate with the table columns and rows,
22 respectively. Visually, the column headers 302 appear just above the columns and
23 are labeled with letters A, B, C, etc., where as the row headers 304 reside to the
24 left of the rows and are labeled with numbers 1, 2, 3, etc.

Smart Selection

When the user selects text inside or outside a table, the architecture intelligently discerns the type of content in the cell or free floating field. For instance, the architecture determines whether the cell or free floating field contains a data value, text, or a formula. If the selected cell or free floating field contains text or a value, the UI exhibits the selection as a character-based cursor ready for cell editing. If the selected cell or free floating field contains a formula, the UI exhibits the selection by highlighting the entire result of the formula. A second selection gesture will allow the user to edit the formula within the formula edit box.

Key Processing

Certain keys have different interpretations depending upon the contents of the cell. This dynamic interpretation accommodates the competing interests of a word processing table and a spreadsheet. As an example, the "Enter" key typically means return in word processing, whereas it means move to the next cell and commit the formula in a spreadsheet program.

If the cell contains text (e.g., cells A1-A6, B1-B5, and C1 in Fig. 3), the architecture interprets this cell as primarily being a word processing-based cell and treats the keys as if the user were working within a word processing application. Thus, an "Enter" key means return, a "tab" key means tab over some distance, the "=" key is typed in anywhere but the beginning of the cell as the equals symbol without denoting a formula, and so forth. The same behavior is applied when the user is editing text within a paragraph or a free floating field.

1 If the cell or free floating field contains a formula or a data value (e.g., cells
2 C2-C6 and FFF1 in Fig. 3), the architecture interprets this cell as primarily being a
3 spreadsheet-based cell and treats the keys as if the user were working within a
4 spreadsheet application. Thus, an "Enter" key and "tab" key mean navigation
5 commands to move to the next cell or move outside the free floating field, the "="
6 key at the beginning of a cell implies the start of a formula, and so forth.

7 8 Table Expansion

9 Spreadsheet users are accustomed to the look and feel of an infinite grid.
10 While the spreadsheet look of Fig. 3 does not have the same infinite grid feel, the
11 table 124 has a column expansion control 306 and a row expansion control 308
12 that allow easy addition of columns and rows, respectively. The controls 306 and
13 308 include an actuatable addition icon together with a dashed column/row to
14 suggest that additional columns and rows may be added by simply actuating the
15 icon.

16 Resize Behavior

17 The table 124 preserves column width and wraps text as the user enters
18 sentences and phrases. In Fig. 3, notice that item 2 in cell B5 wraps within the
19 cell, rather than resizing the column width to accommodate the entire item.

20 21 Formula Edit Box

22 The architecture provides a formula edit box for inserting or editing a
23 formula in a table cell or free floating field. The formula edit box overlays the
24 original cell or free floating field in which the formula resides and initially
25 defaults to the size and shape of the cell or underlying text. If the formula exceeds

1 the initial size, the formula edit box is resized to accommodate the formula.
2 During resizing, the formula edit box initially grows horizontally in length, and
3 then vertically in depth. The underlying table or paragraph does not resize. The
4 ability to resize the local formula edit box, rather than the cell and the table or the
5 paragraph, eliminates a potential UI problem of watching the table or paragraph
6 resize strangely as the user clicks into and out of the cell containing the formula.

7 Examples of the formula edit box are illustrated in Figs. 3, 4, 5 and 6. In
8 Fig. 3, a formula edit box 312 hovers over cell C6 to accept the summation
9 formula. Additionally, formula edit boxes appear over 126(1) and 126(2). In Fig.
10 4, a formula edit box 408 appears for formula 402. In Fig. 5, a formula edit box
11 506 floats above cell C3 and is resized to accommodate the expanded formula.
12 Notice that the underlying table cell C3 and the table as a whole is not resized. In
13 Fig. 6, a formula edit box 610 resides above the free floating field 608 and is
14 resized to hold the long formula.

15 16 Reference Edit

17 The table 124 and free floating field 126 allow the user to perform a
18 reference edit operation, in which the user references one or more cells to extract
19 their data values for inclusion in a formula in another cell or free floating field. In
20 Fig. 3, the user begins entering a summation formula (i.e., "=SUM") in the
21 formula edit box 312 above cell C6. The user then references cells C2 through C5
22 using a pointer 310 or some other mechanism. The referenced cells C2:C5 are
23 highlighted or otherwise indicated as being selected, as represented by the bold
24 rectangular box around the cells.

1 When the user finishes selecting the desired cells (e.g., releasing the left
2 mouse button after highlighting cells C2:C5), references to the cells are added to
3 the summation formula in formula edit bock 312 (i.e., "=SUM(C2:C5)"). Upon
4 further confirmation by the user (e.g., pressing the "Enter" key), an update event is
5 generated and the architecture 100 recalculates the formula and updates the cell C6
6 to display the sum of the cells C2:C5, or \$12,060, in cell C6.

8 Reference Edit for Text

9 Free floating fields provide a unique mechanism for creating references to
10 text via a reference edit operation. In Fig. 4, the user begins entering a formula
11 (i.e., "=") into a formula edit box 408 overlaying a free floating field 402. The
12 user then references the text 404 using a pointer 406 or some other mechanism.
13 The referenced text string is highlighted or otherwise indicated as being selected
14 and a second free floating field is automatically formed around the text range 404.

15 When the user finishes selecting the text range (e.g., releasing the left
16 mouse button after highlighting text 404), a reference to the newly created free
17 floating field is added to the formula in formula edit box 408 (i.e., "=FFF1").
18 Upon further confirmation by the user (e.g., pressing the "Enter" key), an update
19 event is generated and the architecture 100 recalculates the formula and updates
20 the free floating field 402 to display the text "Formal Bid". It is noted that the
21 reference could be made to text within a table cell, to text outside the table or to
22 text inside the table elsewhere in the cell or in another cell.

Cross-Table Referencing and Universal Recalculation

The architecture 100 supports cross-table and cross-field references where a cell in one table or field contains a formula referencing a cell in another table. The architecture 100 also supports cross-table and cross-field reference edit operations that permit a user to reference a cell in one table or a free floating field when entering a formula into another table.

Fig. 6 illustrates cross-table referencing, where table 606 contains references to tables 602 and 604. All three tables are separate and independent from one another, and architecturally have their own set of grid, spreadsheet and table objects 130, 106, and 104. In Fig. 6, cell B2 in table 606 contains a summation formula for adding values in cells B2 and B3 of table 602 (i.e., =SUM(Table1!B2:Table1!B3)). Cell B3 in table 606 contains a summation formula for adding values in cells B2 through B4 in table 604 (i.e., =SUM(Table2!B2:Table2!B4)).

The ability to cross-reference other tables or free floating fields is beneficial in that all tables and free floating fields can be universally updated for any change in just one of the tables. For example, suppose the user changes the value in cell B2 of table 602 from \$300 to \$400. As a result of this change, table 602 is updated to reflect the new value \$400 and the total amount in cell B6 is updated from \$3,060 to \$3,160. Additionally, the value in cell B2 of table 606 is updated to \$2,400, causing the total amount in cell B4 in table 606 to be changed from \$7,800 to \$7,900.

The cross-table referencing is a significant improvement over conventional OLE techniques of embedding a spreadsheet object within a word processing document. With OLE, each spreadsheet object is independent from each other and

cannot reference cells in one another automatically. Since there is no cross-referencing ability, the OLE approach cannot support universal updating throughout the document's spreadsheets as a result of changing a value in one spreadsheet.

Free Floating Field Reference Edit

The reference edit operation and cross-referencing is also available when entering a formula for a free floating field. Consider the document 600 in Fig. 6. The writer is attempting to summarize the total cost of all work items in the opening paragraph. Rather than typing in a hard value, the user decides to insert a free floating field 608 that will hold the total for the job. By using a free floating field 608, the amount can be automatically updated as other estimates in the underlying tables are modified.

Using a reference edit operation, the user can enter the formula in the edit box 610 for free floating field 608 by selecting cell B6 in table 602 to capture element "Table1!B6" and then selecting cell B8 in table 604 to capture element "Table2!B8". When the user confirms this formula, the formula edit box 610 disappears and the total value of "\$12,060" is inserted into the free floating field 608.

In the event the user subsequently changes the estimate of any item in tables 602 or 604, the total value in free floating field 608 is automatically updated. Extending a previous example, suppose the user changes the value in cell B2 of table 602 from \$300 to \$400. As a result of this change, table 602 is updated to reflect the new value \$400 and the total amount in cell B6 is updated from \$3,060 to \$3,160. The value in cell B2 of table 606 is updated to \$2,400,

1 causing the total amount in cell B4 in table 606 to be changed from \$7,800 to
2 \$7,900. Additionally, the total amount in free floating field 608 is updated from
3 \$12,060 to \$12,160. All of the updating throughout the tables and free floating
4 fields is performed automatically in response to the user's change of a single cell
5 in a single table.

6 It is further noted that a free floating field may reference another free
7 floating field. For instance, another free floating field may be added in document
8 600 to reference the first free floating field 608, or a combination of the free
9 floating field 608 and a table cell in tables 602, 604, and 606.

11 Nesting

12 The architecture 100 supports tables and free floating fields nested within
13 one another. Fig. 5 illustrates a situation in which an inner table 504 is nested
14 within a cell B3 of outer table 502. The two tables are independently managed
15 and each has its own underlying pair of grid object 130 and spreadsheet objects
16 106. In this example, cell C3 in outer table 502 is referencing an array of cells
17 B1:B3 in inner table 504, as represented by the summation formula in formula edit
18 box 506. Notice that the reference syntax "Table2!B1" in the formula edit box
19 refers to a separate table and not to cell B3. This is essentially the same cross-
20 table reference edit operation described above, even though the reference is to a
21 table nested within another table cell.

22 Similarly, a free floating field may be nested within a table cell, or, vice
23 versa, a table may be nested within a free floating field. In addition, one or more
24 free floating fields may be nested within one or more other free floating fields.
25

1 Essentially, the architecture and UI supports any nesting arrangement involving
2 tables and free floating fields.

3 The nesting architecture is another feature that is an improvement over
4 conventional table and spreadsheet approaches. OLE mechanisms of embedding a
5 spreadsheet object within a word processing document do not support nested
6 tables.

7 8 Common Document Behaviors

9 The architecture allows common document behaviors for the text body and
10 across the tables. Such functions as spell checking, grammar checking, find, and
11 replace are continuous across table and free floating field boundaries, treating the
12 cell and free floating field contents as if they were part of the document.
13 Moreover, text formatting carries across boundaries. Essentially, any features that
14 are added to modify the text can similarly be applied across a table boundary or
15 free floating field to text inside a table or free floating field. The conventional
16 OLE mechanisms of embedding a spreadsheet object within a word processing
17 document were incapable of supporting these common document behaviors that
18 traversed table boundaries.

19 The benefits of this feature are especially apparent for free floating fields.
20 If a user selects an entire paragraph containing several free floating fields, the user
21 can format the paragraph. The formatting will apply to the normal text as well as
22 to all of the free floating fields within the paragraph.

Functionality Features

This section describes how the architecture functionally supports the user interface features described above, including recalculation, reference edit mechanics, cross-table referencing, the formula edit box, and structure changes to the table. These functionality features are described separately below.

Data v. Presentation

The architecture 100 separates data functions from presentation functions of the integrated table/spreadsheet by employing dual objects per table or free floating field. As shown in Fig. 3, there is one pair of spreadsheet and grid objects for each table or floating field. The grid object 130 maintains the data and format information, and facilitates the recalculation process. The corresponding spreadsheet objects 106 and free floating field objects 107 are more concerned with presenting the table and free floating field as part of the document, as well as capturing user inputs into the table and free floating field.

The separation is beneficial because it allows the architecture to support cross-table referencing, reference editing to other tables, and universal recalculation throughout the document. The underlying grid objects do not care how the tables or free floating fields are laid out or where they appear in the document, nor do they care if there are one or multiple grid objects. Similarly, the spreadsheet objects 106 and free floating field objects 107 do not worry about the data and formulas, or the recalculation process.

Recalculation

Recalculation is a function performed by the architecture 100 in response to modification of a table or free floating field. When a modification is made, the architecture 100 recalculates the various formulas in all tables and free floating fields in the document that may have been affected by user input.

Continuing the example of Fig. 6, when the user changes the value in cell B2 of first table 602 from \$300 to \$400, the recalculation engine 142 in spreadsheet engine 112 recalculates the formulas in cell B6 of first table 602 to update the amount from \$3,060 to \$3,160. The recalculation engine 142 also re-computes the formula in cell B2 of third table 606 to yield \$2,400 and the formula in cell B4 in third table 606 to yield \$7,900. Finally, the recalculation engine 142 recalculates the formula in free floating field 608 to update the value from \$12,060 to \$12,160. This recalculation occurs automatically across the entire document in response to the user input.

Fig. 9 illustrates the recalculation process 900 for a single table in more detail. An input version of the user interface table 902(1) is shown at a time when the user enters a new value "7" in cell A1, but has not yet confirmed the entry (e.g., by hitting the "Enter" key or clicking out of the cell). An output version of the UI table 902(2) is shown at a time just after user confirmation.

A corresponding pair of spreadsheet and grid objects 106 and 130 exists for the table 902. The grid object 130 maintains a cell table 134 and a format table 132. Prior to user entry of "7" into cell A1, cell table 134 contains a value "1" in cell A3, a formula referencing cell A1 (i.e., "=A1") in cell C1, and a formula summing cells A1 and C1 (i.e., "=A1+C1") in cell C3. The format table 132 indicates that cell A3 is formatted as a number, and that cells C1 and C3 are

1 formatted as currency in U.S. dollars. The input version of UI table 902(1) shows
2 results of the formatted formulas as \$0.00 in cell C1 and \$1.00 in cell C3.

3 Now, suppose the user enters the value "7" into cell A1 of UI table 902(1),
4 as indicated by the pointer 904. The value is received at the spreadsheet objects
5 106, as indicated by flow arrow 910. Once the user confirms this entry by moving
6 the selection out of the cell A1, the newly entered value "7" is passed to the
7 spreadsheet engine 112 and particularly, the parser 144 of formula manager 140
8 (flow arrow 912).

9 The parser 144 parses the entry and determines it to be a data value. The
10 parser 144 puts the data value into cell A1 of the cell table 134 (flow arrow 914).
11 This insertion causes a table change event, which is sent to the recalculation
12 engine 142 to initiate a recalculation (flow arrow 916). The recalculation engine
13 142 runs through the formula chain to recalculate any formula anywhere that is
14 affected by the new data value in cell A1. In this case, the formulas in cells C1
15 and C3 are affected and hence, these formulas are recalculated (flow arrow 918).
16 The recalculation produces a result of "7" in cell C1 and a result of "8" in cell C3.

17 Afterwards, the format table 132 is consulted to determine the desired
18 format for the new value and recalculated formulas (flow arrow 920). Here, the
19 formula results are formatted as currency as indicated by the "\$" symbols in cells
20 C1 and C3, and the new value "7" is formatted as a number as indicated by the "#"
21 symbol in cell A1.

22 The spreadsheet engine returns the formatted results \$7.00, \$8.00, and 7 to
23 the spreadsheet objects 106 (flow arrow 922). The spreadsheet objects 106
24 updates the table with these formatted results to produce the output version of the
25 UI table 902(2) (flow arrow 924).

1 The recalculation event is essentially instantaneous. The user merely sees
2 an immediate change in the UI table from input version 902(1) to output version
3 902(2).

4 The above example of recalculation was described in the context of tables.
5 However, it is noted that essentially the same recalculation operations may be
6 performed in free floating fields, as is described below in more detail.

8 Reference Edit Mechanics

9 The reference edit mechanism allows the user to reference another cell to
10 obtain data, rather than forcing the user to type in a value or the reference syntax.
11 In Fig. 9, consider the situation when the user created the formula “=A1” in cell
12 C1. The user selects cell C1, types in an “=” sign to indicate a formula, and then
13 references the cell A1 by moving the mouse pointer 904 to cell A1 and clicking.
14 The spreadsheet objects 106 (namely, CellEditing object 152) capture this
15 reference and pass it to parser 144. The parser 144 recognizes it as a formula,
16 creates a formula object and inserts the formula into a cell of cell table 134, as
17 indicated by cell C1. Again, a similar mechanism may be used for free floating
18 fields.

20 Cross-Table and Field Referencing and Universal Recalculation

21 With architecture 100, reference editing may be extended across multiple
22 tables and free floating fields distributed throughout a document. A cell in one
23 table or a free floating field may reference a cell in another table, a different free
24 floating field, or a combination of a table cell and free floating field. The
25

1 architecture 100 automatically recalculates all tables and free floating fields that
2 are affected by a change in any one table cell or free floating field.

3 Fig. 10 illustrates the recalculation process 1000 for a document 1002
4 containing multiple tables 1004(1),..., 1004(N) and multiple free floating fields
5 1006(1), ..., 1006(M) distributed throughout the text body. A corresponding set of
6 spreadsheet, free floating field, grid objects 106, 107 and 130 is created for each
7 table 1004(1)-1004(N) and each free floating field 1006(1)-1006(M) in the
8 document 1002.

9 In this illustration, spreadsheet object 106(1) and associated grid object
10 130(1) support UI table 1004(1), spreadsheet object 106(N) and associated grid
11 object 130(N) support UI table 1004(N), free floating field object 107(1) and
12 associated grid object 130(N+1) support free floating field 1006(1), and free
13 floating field object 107(M) and associated grid object 130(N+M) support free
14 floating field 1006(M). The table grid objects 130(1)-130(N) each contain a cell
15 table 134(1)-134(N) and a format table 132(1)-132(N). The FFF grid objects
16 130(N+1)-130(N+M) each contain a single cell 136(1)-136(M) and a
17 corresponding format cell 138(1)-138(M).

18 Suppose the user is entering a summation formula in cell B1 of UI table
19 1004(N) that adds three cells C1-C3 in table 1004(1). Rather than typing in the
20 reference syntax (i.e., "=SUM(Table1!C1:Table1!C3)"), the user may simply
21 move the pointer 1010 to table 1004(1) and select the desired array of cells, as
22 indicated by the selection block 1012. The spreadsheet object 106(N) (namely, the
23 CellEditing object) associated with the source table 1004(N) recognizes the
24 reference edit operation and captures the selected cells C1-C3 in remote
25 referenced table 1 (flow arrow 1020). When the user confirms this entry by

moving the selection out of the referenced table 1004(1), the newly entered formula “=SUM(Table1!C1:Table1!C3)” is passed to the spreadsheet engine 112 and particularly, the parser 144 of formula manager 140 (flow arrow 1022).

The parser 144 determines that the entry is a formula and creates a formula object (not shown) and adds the formula to the formula chain. The parser 144 puts the formula into cell B1 of the cell table 134(N) in cell table N (flow arrow 1024). This insertion generates a table change event, which is sent to the recalculation engine 142 to initiate a recalculation (flow arrow 1026).

The recalculation engine 142 runs through the entire formula chain to recalculate any formula in any table or free floating field that is affected by adding the new formula in cell B1. In this case, the formulas in free floating cells 136(1) and 136(M) are affected. But, since these formulas rely on the result of the newly entered formula in table N, they are moved to the end of the formula chain. Thus, the new formula is first calculated (flow arrow 1028), and then the formulas in free floating field cells 136(1) and 136(M) are recalculated (flow arrows 1030 and 1032). The recalculation produces a result of “425” for table cell B1 in cell table N, a result of “425” in FFF cell 136(1), and a result of “425” in FFF cell 136(M).

Afterwards, the various format tables 132(N) and format cells 138(1) and 138(M) are consulted to determine the desired format for the results of the recalculated formulas. Here, all formula results are formatted as currency as indicated by the “\$”. The spreadsheet engine 112 returns the formatted results “\$425.00” to the associated spreadsheet objects 106(N), 107(1), and 107(M) (flow arrows 1034, 1036, 1038). The spreadsheet objects then update their associated table and free floating fields with these formatted results to produce the output as shown in Fig. 10 (flow arrows 1040, 1042, 1044).

1 Once again, the recalculation event is essentially instantaneous and the user
2 merely perceives an immediate change in the affected UI table and free floating
3 fields throughout the document 1002.

4 5 Formula Edit Box

6 The CellEditing objects 152 and 162 manage the formula edit box that
7 permits user edits of formulas in table cells and free floating fields. The formula
8 edit box is provided in the user interface in response to the user entering the “=”
9 symbol at the beginning of a cell. The formula edit box is overlaid as a separate
10 entry field above the cell or free floating field into which the user is inserting a
11 formula. The CellEditing object 152 or 162 captures the user entry of various
12 variants in the formula, and facilitates reference editing to other cells or free
13 floating fields as a way to input values. When the formula is entered via the
14 formula edit box and confirmed, the CellEditing object 152 or 162 passes the
15 formula to the formula manager 140 in the spreadsheet engine 112 for parsing.
16 The formula edit box is then removed from the user interface.

17 18 Structural Changes

19 The Table object 104 manages and monitors the user input for structure
20 changes, such as insertion/deletion of a row, merging cells, and so forth. When the
21 user makes a structure change, the Table object 104 fires events to the
22 GridBehavior object 150, which informs the spreadsheet engine 112, and in turn
23 updates the cell table 134 associated with the UI table.

24 As above, any change to the cell table causes an event that is returned to the
25 recalculation engine 142 to initiate a recalculation cycle. The recalculation engine

1 steps through the chain of formulas and updates all cells affected by the structural
2 change, returning any errors that might arise from the structure change (e.g., loss
3 of a reference value as a result of deleting a row/column). The spreadsheet engine
4 then outputs the results of the recalculation and the UI table is updated to reflect
5 the structural change and the recalculation results.

6 7 Cut, Copy, Paste

8 A separate document object may be employed to manage operations
9 affecting the entire document, rather than a specific table or free floating field.
10 The document object plays a role in initially inserting the table/spreadsheet or free
11 floating field into the document. The document object may be constructed
12 similarly to the GridBehavior object 150 and configured to monitor for cut, copy,
13 and paste operations. When a cut or copy operation is performed, the object
14 places the HTML code on the clipboard. Upon a subsequent paste operation, the
15 HTML code is retrieved from the clipboard and inserted into the appropriate
16 location. It is noted that, unlike some spreadsheet programs, the user is not forced
17 to cut/copy and then *immediately* paste.

18 One unique problem that is not encountered by traditional spreadsheet
19 programs is the ability to create a new table or free floating field through a paste
20 operation. When a new table or free floating field is created, the architecture
21 automatically renames the new table or free floating field and adjusts all
22 references within the table or free floating field that was pasted.

1 **Conclusion**

2 Although the description above uses language that is specific to structural
3 features and/or methodological acts, it is to be understood that the invention
4 defined in the appended claims is not limited to the specific features or acts
5 described. Rather, the specific features and acts are disclosed as exemplary forms
6 of implementing the invention.

7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

CLAIMS

We claim:

1. A method comprising:
presenting a free floating field in line with text in a document, the free floating field presenting content derived from a source; and
upon modification of the source, automatically updating the content in the free floating field.
2. The method of claim 1, wherein the document is written in a markup language.
3. The method of claim 1, wherein the source is text and the free floating field presents the text.
4. The method of claim 1, wherein the source is a data value and the free floating field presents the data value.
5. The method of claim 1, wherein the source resides separate from the free floating field and the free floating field presents content derived from referencing the source.
6. The method of claim 1, further comprising:
presenting a table containing multiple cells in which one cell is the source;
and

1 upon modification of the cell, automatically updating the content in the free
2 floating field.

3
4 7. The method of claim 1, wherein the free floating field is a first free
5 floating field, the method further comprising:

6 presenting a second free floating field, the second free floating field
7 presenting content derived from referencing the first free floating field; and
8 upon modification of the source, automatically updating the contents in the
9 first and second free floating fields.

10
11 8. The method of claim 1, further comprising overlaying a formula edit
12 box on the free floating field to facilitate user entry of a formula into the free
13 floating field.

14
15 9. The method of claim 8, further comprising resizing the formula edit
16 box as the user enters the formula.

17
18 10. The method of claim 8, further comprising extending the formula
19 edit box horizontally and subsequently vertically as the user enters the formula.

20
21 11. The method of claim 1, further comprising:
22 creating a cell structure in association with the free floating field, the cell
23 structure holding one of a formula or data used in the free floating field; and
24 creating a format structure in association with the free floating field, the
25 format structure holding formatting information for the free floating field.

1
2 **12.** The method of claim 1, further comprising formatting the free
3 floating field independently of the text.

4
5 **13.** The method of claim 1, further comprising modifying a format of
6 the text and automatically applying the format modification to the free floating
7 field.

8
9 **14.** The method of claim 1, further comprising:
10 determining, upon selection of the free floating field, a type of content in
11 the free floating field; and
12 interpreting user entry based upon the type of content in the free floating
13 field.

14
15 **15.** The method of claim 14, wherein the determining comprises:
16 evaluating whether the type of content is a formula or non-text data;
17 if the type of content is a formula or non-text data, interpreting the user
18 entry as applicable to spreadsheet functions; and
19 if the type of content is not a formula or non-text data, interpreting the user
20 entry as applicable to word processing functions.

21
22 **16.** The method of claim 14, wherein the determining comprises:
23 evaluating whether the type of content is a formula;
24 if the type of content is a formula, highlighting all of the formula and
25 allowing editing in a formula edit box; and

1 if the type of content is not a formula, placing a cursor in the free floating
2 field.

3
4 **17.** A computer readable medium having computer-executable
5 instructions that, when executed on one or more processors, perform the method as
6 recited in claim 1.

7
8 **18.** A method comprising:
9 inserting a free floating field in text in a document;
10 enabling a user to enter a formula into the free floating field; and
11 automatically recalculating the formula in the free floating field.

12
13 **19.** The method of claim 18, wherein the enabling comprises overlaying
14 a formula edit box on the free floating field to facilitate user entry of the formula
15 into the free floating field.

16
17 **20.** The method of claim 19, further comprising resizing the formula
18 edit box as the user enters the formula.

19
20 **21.** The method of claim 19, further comprising extending the formula
21 edit box horizontally and subsequently vertically as the user enters the formula.

1 **22.** The method of claim 18, further comprising inserting a table that
2 contains multiple cells, wherein the data value resides in one cell of the table so
3 that upon modification of the data value in the cell, the formula in the free floating
4 field is automatically recalculated.

5
6 **23.** The method of claim 18, wherein the free floating field is a first free
7 floating field and the formula is a first formula, the method further comprising:
8 inserting a second free floating field;
9 enabling the user to enter a second formula into the second free floating
10 field, the second formula referencing the first free floating field; and
11 upon modification of the data value, the first and second formulas in the
12 first and second free floating fields are automatically recalculated.

13
14 **24.** The method of claim 18, further comprising:
15 creating a cell structure in association with the free floating field, the cell
16 structure holding the formula; and
17 creating a format structure in association with the free floating field, the
18 format structure holding formatting information for the free floating field.

19
20 **25.** A computer readable medium having computer-executable
21 instructions that, when executed on one or more processors, perform the method as
22 recited in claim 18.

23
24 **26.** A method comprising:
25 presenting a free floating field in line with text; and

1 overlaying a formula edit box on the free floating field to facilitate user
2 entry of a formula into the free floating field.

3
4 **27.** The method of claim 26, wherein the formula edit box initially
5 defaults to a size and shape of the free floating field.

6
7 **28.** The method of claim 26, further comprising resizing the formula
8 edit box as the user enters the formula.

9
10 **29.** The method of claim 26, further comprising extending the formula
11 edit box horizontally and subsequently enlarging the formula edit box vertically as
12 the user enters the formula.

13
14 **30.** The method of claim 26, further comprising:
15 presenting at least one table; and
16 enabling a user to reference a cell in the table to add a data value to the
17 formula.

18
19 **31.** A computer readable medium having computer-executable
20 instructions that, when executed on one or more processors, perform the method as
21 recited in claim 26.

22
23 **32.** A method comprising:
24 presenting a free floating field in line with text; and
25

1 presenting a table within the document, the table having a cell with
2 contents; and

3 enabling a user to reference the cell in the table when entering a formula in
4 the free floating field.

5
6 **33.** The method of claim 32, further comprising, upon modification of
7 the contents in the cell of the table, automatically recalculating the formula in the
8 free floating field.

9
10 **34.** The method of claim 32, further comprising nesting the free floating
11 field within a cell of the table.

12
13 **35.** The method of claim 32, further comprising nesting the table within
14 the free floating field.

15
16 **36.** The method of claim 32, wherein the formula also references a value
17 outside of the table.

18
19 **37.** A computer readable medium having computer-executable
20 instructions that, when executed on one or more processors, perform the method as
21 recited in claim 32.

22
23 **38.** A method comprising:
24 inserting one of a first free floating field or a spreadsheet table in text;
25

1 in response to user selection of at least a portion of the text or data,
2 automatically creating a second free floating field containing the portion of the
3 text; and

4 creating a reference in the first free floating field or spreadsheet table to the
5 second free floating field.

6
7 **39.** The method of claim 38, further comprising, upon confirmation,
8 displaying the portion of the text in place of the first free floating field.

9
10 **40.** The method of claim 38, further comprising, upon modification of
11 the text in the second free floating field, automatically updating the first free
12 floating field.

13
14 **41.** A computer readable medium having computer-executable
15 instructions that, when executed on one or more processors, perform the method as
16 recited in claim 38.

17
18 **42.** A method comprising:
19 presenting free floating field in line with text in a document;
20 creating a cell structure to hold one of data or a formula for the free floating
21 field;
22 creating a format structure to hold formatting information for the free
23 floating field;
24 receiving, into the free floating field, user entry of a reference to a source in
25 the document;

1 parsing the user input to update the cell structure and the format structure;
2 in an event the user input causes changes in the cell structure or format
3 structure, updating the cell structure or format structure to produce a new result;
4 and
5 presenting the free floating field with the new result.
6

7 **43.** A computer readable medium having computer-executable
8 instructions that, when executed on one or more processors, perform the method as
9 recited in claim 42.
10

11 **44.** A method comprising:
12 integrating text and a free floating field within a common document, the
13 free floating field supporting spreadsheet functionality;
14 enabling a user to select a control function to modify or evaluate an aspect
15 of the document; and
16 applying the control function across both the text and the free floating field.
17

18 **45.** The method of claim 44, wherein the control function is selected
19 from a group of functions including formatting, spell checking, grammar
20 checking, find, find and replace, auto-correct, applying document themes,
21 inserting lists, images, drawings, charts, hyperlinks, automatic detection of
22 hyperlinks, and automatic detection of lists.
23
24
25

1 **46.** The method of claim 44, wherein the control function is any text
2 feature that can be applied to the text and the applying comprises applying that
3 text feature to the free floating field.
4

5 **47.** A method comprising:
6 presenting a free floating field in line with text; and
7 number formatting the free floating field independent of the text.
8

9 **48.** A user interface comprising:
10 a text entry area that permits entry of individual lines of text; and
11 a free floating field residing inline within a line of text, the free floating
12 field presenting content derived from source data or referencing source data such
13 that upon modification of the source data, the free floating field automatically re-
14 derives the content and presents the re-derived content.
15

16 **49.** The user interface of claim 48, wherein the content of the free
17 floating field is presented as text when not being edited.
18

19 **50.** The user interface of claim 48, wherein the free floating field
20 exhibits a change in appearance when selected for editing.
21

22 **51.** The user interface of claim 48, wherein the free floating field
23 contains a formula and the source is a data value.
24
25

1 **52.** The user interface of claim 48, wherein a formula edit box is
2 overlaid on the free floating field to facilitate user entry of a formula into the free
3 floating field.

4
5 **53.** The user interface of claim 52, wherein the formula edit box
6 dynamically resizes as the user enters the formula.

7
8 **54.** The user interface of claim 52, wherein the formula edit box extends
9 horizontally and subsequently enlarges vertically as the user enters the formula.

10
11 **55.** The user interface of claim 48, wherein the free floating field is a
12 first free floating field and further comprising a second free floating field inline
13 within text, the second free floating field presenting content derived from
14 referencing the first free floating field.

15
16 **56.** The user interface of claim 48, further comprising a table with
17 multiple cells, the table having a particular cell that presents content derived from
18 referencing the free floating field.

19
20 **57.** A user interface comprising:
21 at least one table residing within a document, the table having multiple
22 cells;
23 at least one free floating field inline with text in the document, the free
24 floating field containing a formula that references a cell in the table; and
25

1 the formula in the free floating field being automatically recalculated upon
2 modification of the cell in the table.

3
4 **58.** The user interface of claim 57, wherein the free floating field is a
5 first free floating field and further comprising a second free floating field
6 containing a reference to the first free floating field.

7
8 **59.** The user interface of claim 57, wherein the free floating field is
9 nested within a cell of the table.

10
11 **60.** The user interface of claim 57, wherein the table is nested within the
12 free floating field.

13
14 **61.** The user interface of claim 57, further comprising a formula edit
15 box overlaid on the free floating field to facilitate user entry of the formula.

16
17 **62.** The user interface of claim 57, further comprising multiple tables
18 and multiple free floating fields, at least one of the tables and free floating fields
19 containing a formula that references at least one other of the tables and free
20 floating fields.

21
22 **63.** An architecture comprising:
23 a user interface to present text and a free floating field inline with the text;
24 a free floating field component to receive data or a formula entered into the
25 free floating field; and

1 a spreadsheet functionality manager to manage spreadsheet functions for
2 the free floating field.

3
4 **64.** The architecture of claim 63, wherein the user interface overlays a
5 formula edit box on the free floating field to facilitate user entry of a formula.

6
7 **65.** The architecture of claim 63, wherein the spreadsheet functionality
8 manager comprises:

9 a cell structure to maintain the data or formula entered into the free floating
10 field; and

11 a format table to maintain formatting information used in the free floating
12 field.

13
14 **66.** The architecture of claim 63, wherein the free floating field receives
15 a formula, and the spreadsheet functionality manager comprises:

16 a first memory structure to hold source data;

17 a second memory structure to maintain the formula entered into the free
18 floating field, the formula referencing the data in the first memory structure; and

19 a recalculation engine to recalculate the formula in the second memory
20 structure following a change to the data in the first memory structure.

21
22 **67.** The architecture of claim 63, wherein the user interface presents at
23 least one table within the document, and the spreadsheet functionality manager is
24 configured to track references made between the free floating field and the table.
25

1 **68.** The architecture of claim 63, wherein the user interface presents at
2 least one table within the document, and the spreadsheet functionality manager is
3 configured to track references made between the free floating field and the table,
4 the spreadsheet functionality manager being further configured to update any data
5 and formulas in the table and free floating fields that are affected by a change
6 made to one of the table or the free floating field.

7
8 **69.** A computer comprising:
9 a memory;
10 a processing unit coupled to the memory; and
11 an architecture stored in the memory and executable on the processing unit
12 to construct and display a document having a free floating field inline with text,
13 the free floating field supporting spreadsheet functionality.

14
15 **70.** The computer of claim 69, wherein the free floating field contains a
16 formula that references source data, and upon modification of the source data, the
17 architecture automatically recalculates the formula in the free floating field.

18
19 **71.** The computer of claim 69, wherein the architecture constructs
20 multiple free floating fields within the document, at least one free floating field
21 containing a reference to contents in another free floating field.

1 72. The computer of claim 69, wherein the architecture constructs a
2 table within the document and the free floating field contains a reference to
3 contents in the table.
4

5 73. The computer of claim 69, wherein the architecture comprises:
6 a user interface manager to receive user input into the free floating field;
7 and
8 a spreadsheet functionality manager to manage the spreadsheet
9 functionality of the free floating field based on the user input received by the user
10 interface manager.
11

12 74. A computer readable medium having computer-executable
13 instructions that, when executed on one or more processors, performs the
14 following:

15 display a free floating field in line with text in a document;
16 create a reference, within the free floating field, to at least one source
17 elsewhere in the document; and
18 upon modification of the source, automatically update the free floating
19 field.
20

21 75. The computer readable medium of claim 74, wherein the source is
22 text and the free floating field references the text.
23
24
25

1 76. The computer readable medium of claim 74, wherein the source is a
2 data value and the free floating field contains a formula that references the data
3 value.
4

5 77. The computer readable medium of claim 74, wherein the free
6 floating field is a first free floating field, and further comprising computer-
7 executable instructions to:

8 display a second free floating field;

9 create a reference, within the second free floating field, to the first free
10 floating field; and

11 upon modification of the source, automatically update the first and second
12 free floating fields.
13

14 78. A computer readable medium having computer-executable
15 instructions that, when executed on one or more processors, performs the
16 following:

17 present a free floating field in line with text;

18 receive user-entered formula into the free floating field, the formula
19 referencing at least one data value elsewhere in the document; and

20 upon modification of the data value, automatically recalculate the formula
21 in the free floating field.
22
23
24
25

1 **79.** The computer readable medium of claim 78, further comprising
2 computer-executable instructions to overlay a formula edit box on the free floating
3 field to facilitate user entry of the formula into the free floating field.

4
5 **80.** The computer readable medium of claim 78, further comprising
6 computer-executable instructions to:

7 present a table; and

8 create a reference from the free floating field to a cell in the table.

ABSTRACT

A system architecture integrates spreadsheet functionality into text. The architecture allows insertion of discrete individual fields, referred to as “free floating fields”, inline with normal textual sentences. In an HTML document, for example, the free floating fields are elements constructed along with text elements and rendered together as an integrated document. Once rendered, the free floating fields present contents that resemble normal text consistent with the surrounding text. The free floating fields offer spreadsheet functionality, including the ability to handle complex formulas, reference values in a separate free floating field or table, and automatically recalculate the formulas when a source value changes. The values and formula results can also be formatted (e.g., numbers, date, times, currency, etc.), like a spreadsheet, while remaining part of the normal text of a document. Underlying the user interface, the architecture separates data handling functions from presentation functions. The architecture includes a user interface manager to manage how the free floating fields appear in a document (e.g., selection, cut, copy, paste, etc.) and to facilitate user entry of formulas and values into the fields. The architecture also has a spreadsheet functionality manager to manage the spreadsheet functions for the free floating fields, such as recalculation, formula handling, sorting, referencing, and the like.

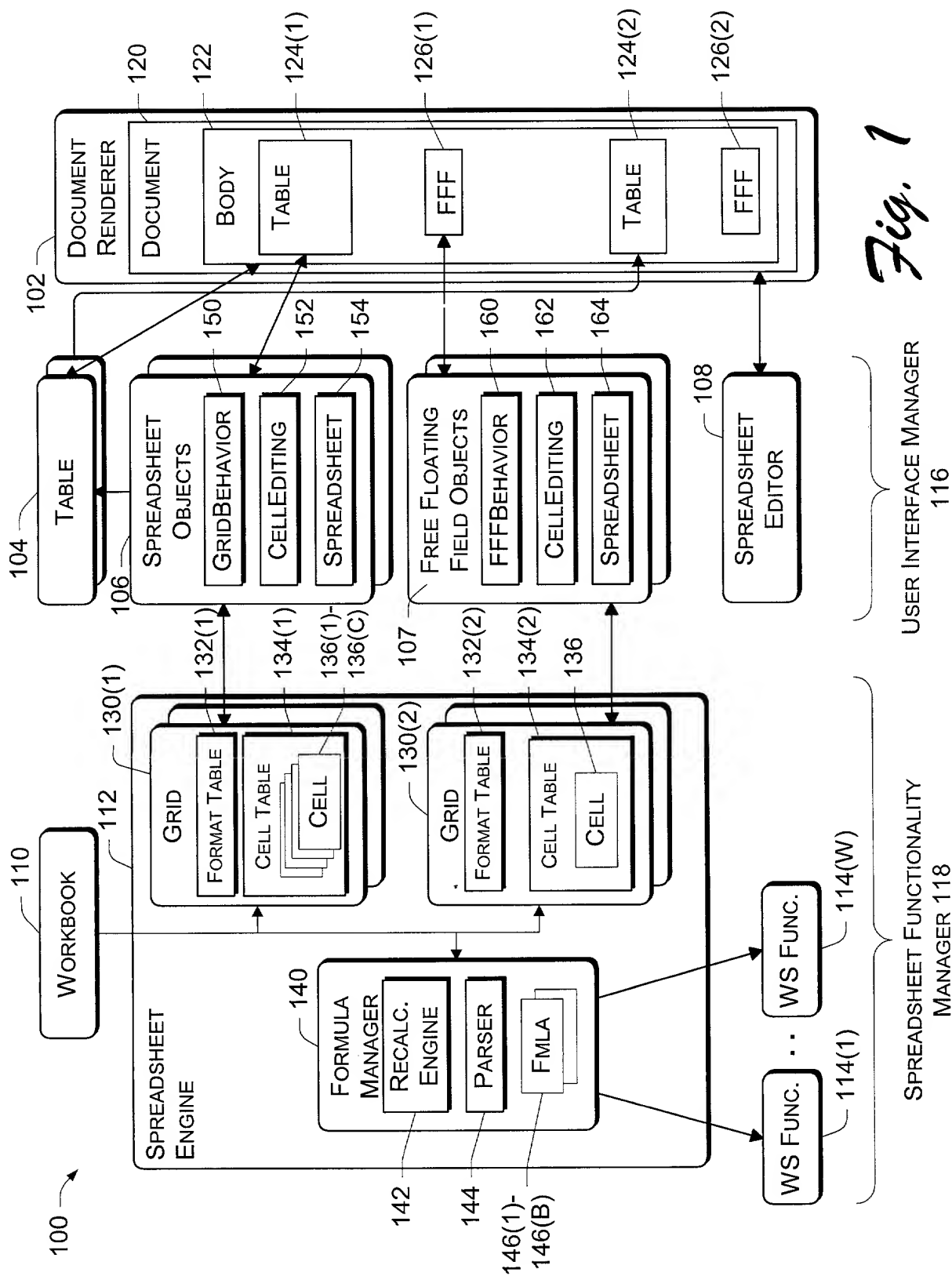


Fig. 1

120

John's Construction
1234 N. Road
Anywhere, WA 99999

Dear Mr. Jones,

Here are my preliminary estimates for the work items. The total cost of the project is \$12,060. If you want to wait on the kitchen repair, the total cost would be reduced to \$6,560.

204(1)	202(1)	202(2)	202(3)	124
	Home Section	Work Items	Cost	
204(2)	Roof	1. Rip off shingles 2. Resurface and apply stucco	\$2,300	
204(3)	Kitchen	1. Repaint Cabinets 2. Rework plumbing for sink 3. Add ceramic tiling to floor	\$5,500	
204(4)	Family Room	1. Fix blinds 2. Reupholster couch 3. Install hardwood floors	\$3,500	
204(5)	Porch	1. Repair structural damage 2. Sand and repaint with ivory Asian paint	\$760	

Thanks,

John

Fig. 2

120

John's Construction
1234 N. Road
Anywhere, WA 99999

Dear Mr. Jones,

Here are my preliminary estimates for the work items. The total cost of the project is $\text{=SUM(Table1!C2:C5)}$ 126(1) Wait on the kitchen repair, the total cost would be reduced to =FFF1-Table1!C3 126(2)

	302(1)	302(2)	302(3)	124
304(1)	A	B	C	306
1	Home Section	Work Items	Cost	
304(2)	2	Roof	1. Rip off shingles 2. Resurface and apply stucco	\$2,300
304(3)	3	Kitchen	1. Repaint Cabinets 2. Rework plumbing for sink 3. Add ceramic tiling to floor	\$5,500
304(4)	4	Family Room	1. Fix blinds 2. Reupholster couch 3. Install hardwood floors	\$3,500
304(5)	5	Porch	1. Repair structural damage 2. Sand and repaint with ivory Asian paint	\$760
304(6)	6	Total	$\text{=SUM($	310

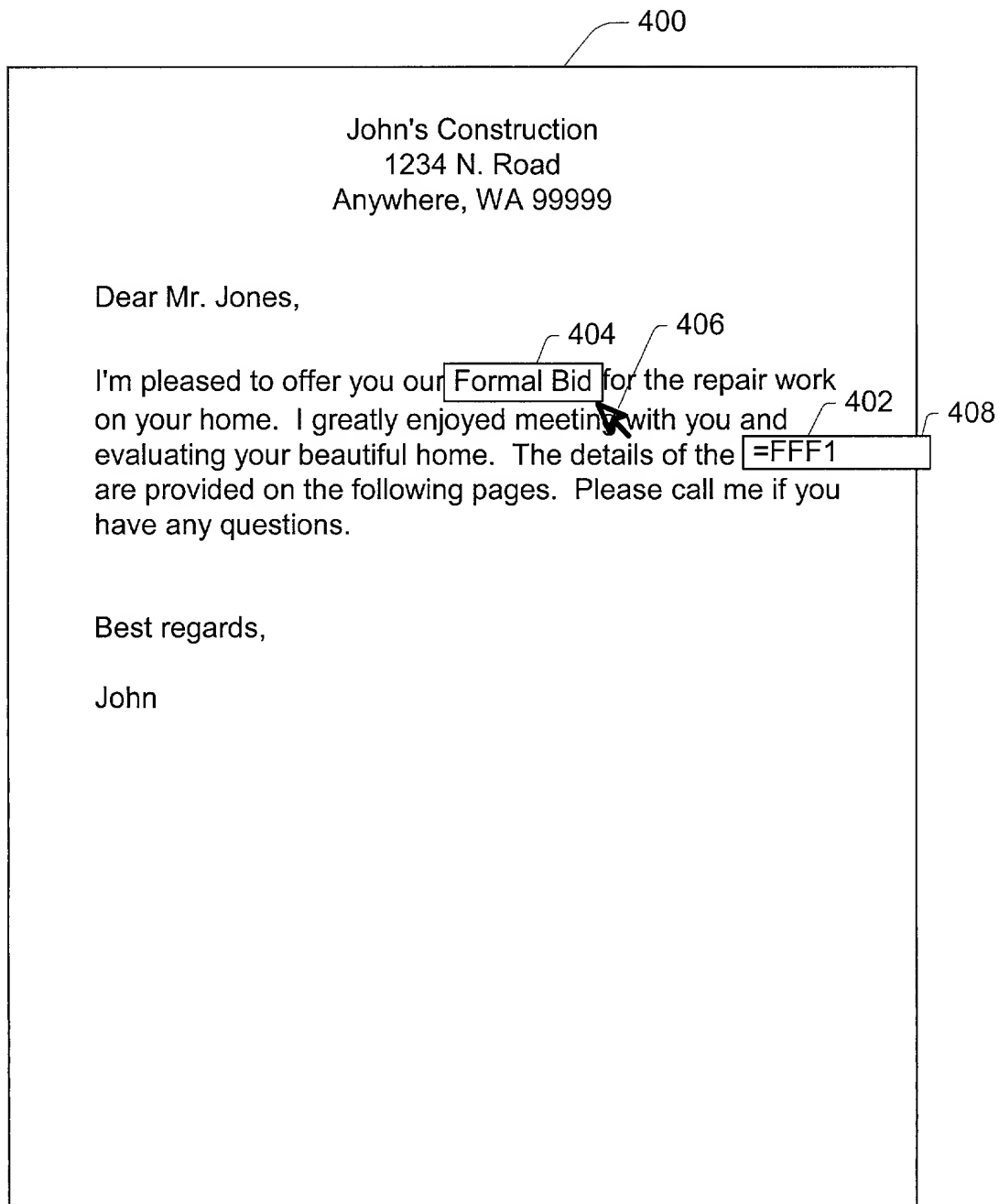
308

Thanks,

John

312

Fig. 3

*Fig. 4*

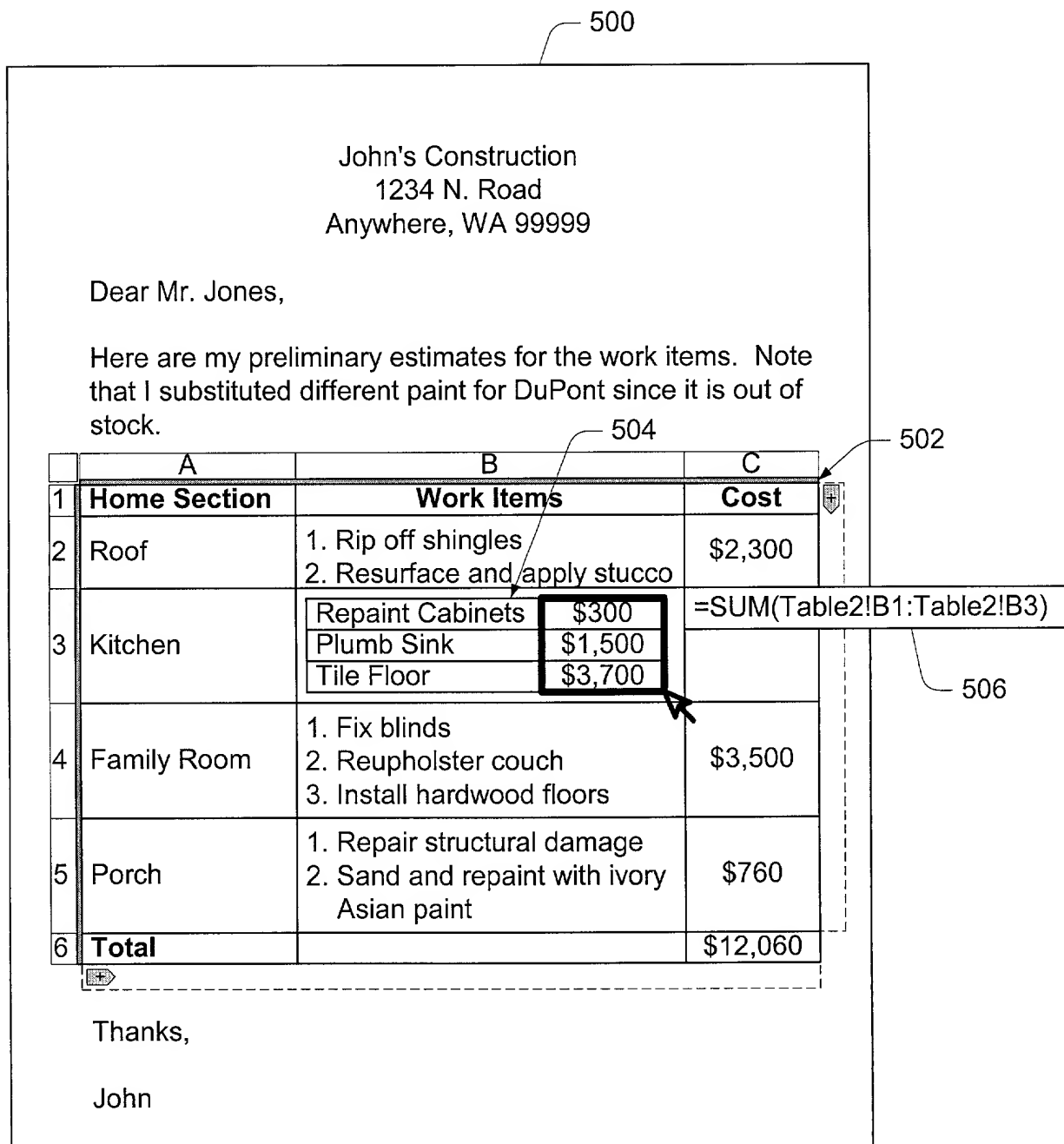


Fig. 5

600

John's Construction
1234 N. Road
Anywhere, WA 99999

Dear Mr. Jones,

Here are my preliminary estimates for the work items. The total cost is approximately `=Table1!B6+Table2!B8` 610

602 → Table 1

Outdoor Work Items	Cost
1. Rip off roof shingles	\$300
2. Resurface and stucco	\$2,000
3. Repair porch damage	\$360
4. Paint porch	\$400
Total	\$3,060

608 → The itemized cost for the outdoor work items are summarized in Table 1. Note that I substituted different paint for DuPont since it is out of stock.

604 → Table 2

The itemized cost for the indoor work items are summarized in Table 2.

Indoor Work Items	Cost
1. Repaint Cabinets	\$300
2. Plumb sink	\$1,500
3. Tile floor	\$3,700
4. Fix blinds	\$200
5. Reupholster couch	\$400
6. Install hardwood floors	\$2,900
Total	\$9,000

As you requested, the costs of pursuing only the roof and kitchen at this time are summarized in Table 3.

606 →

Big Items Only	Cost
Roof	\$2,300
Kitchen	\$5,500
Total	\$7,800

`=SUM(Table1!B2:Table1!B3)`

`=SUM(Table2!B2:Table2!B4)`

Table 3

Thanks, John

Fig. 6

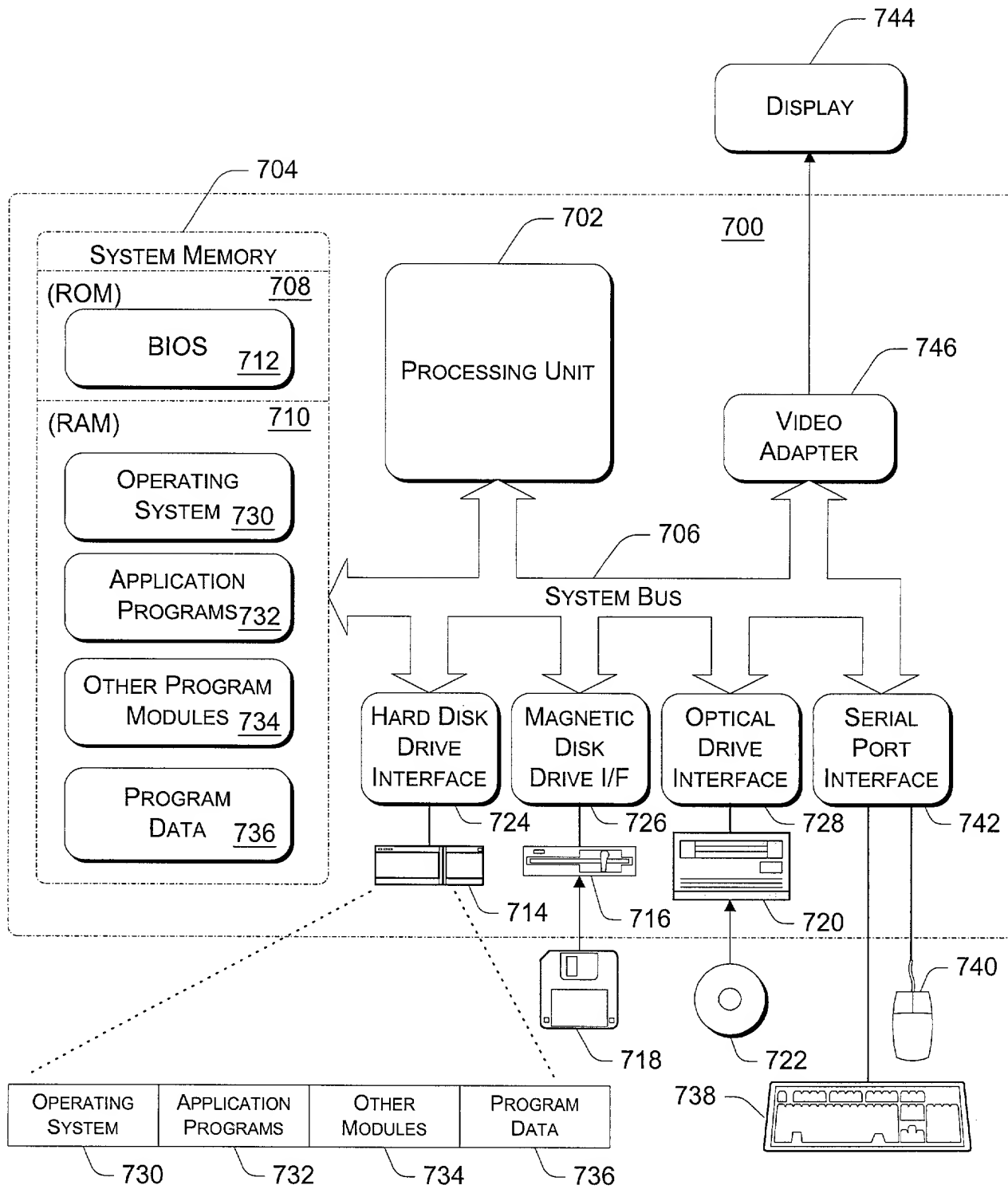


Fig. 7

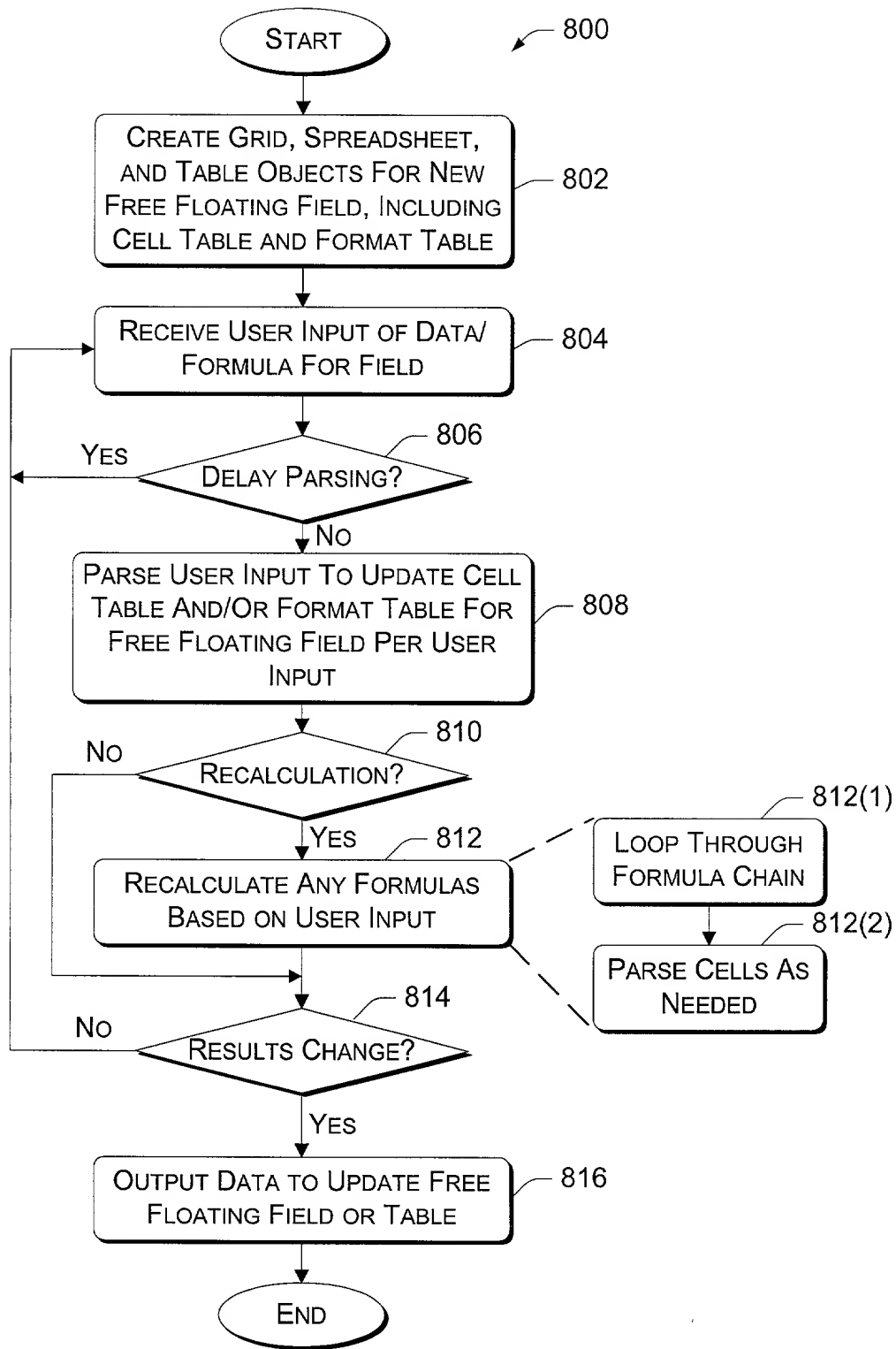


Fig. 8

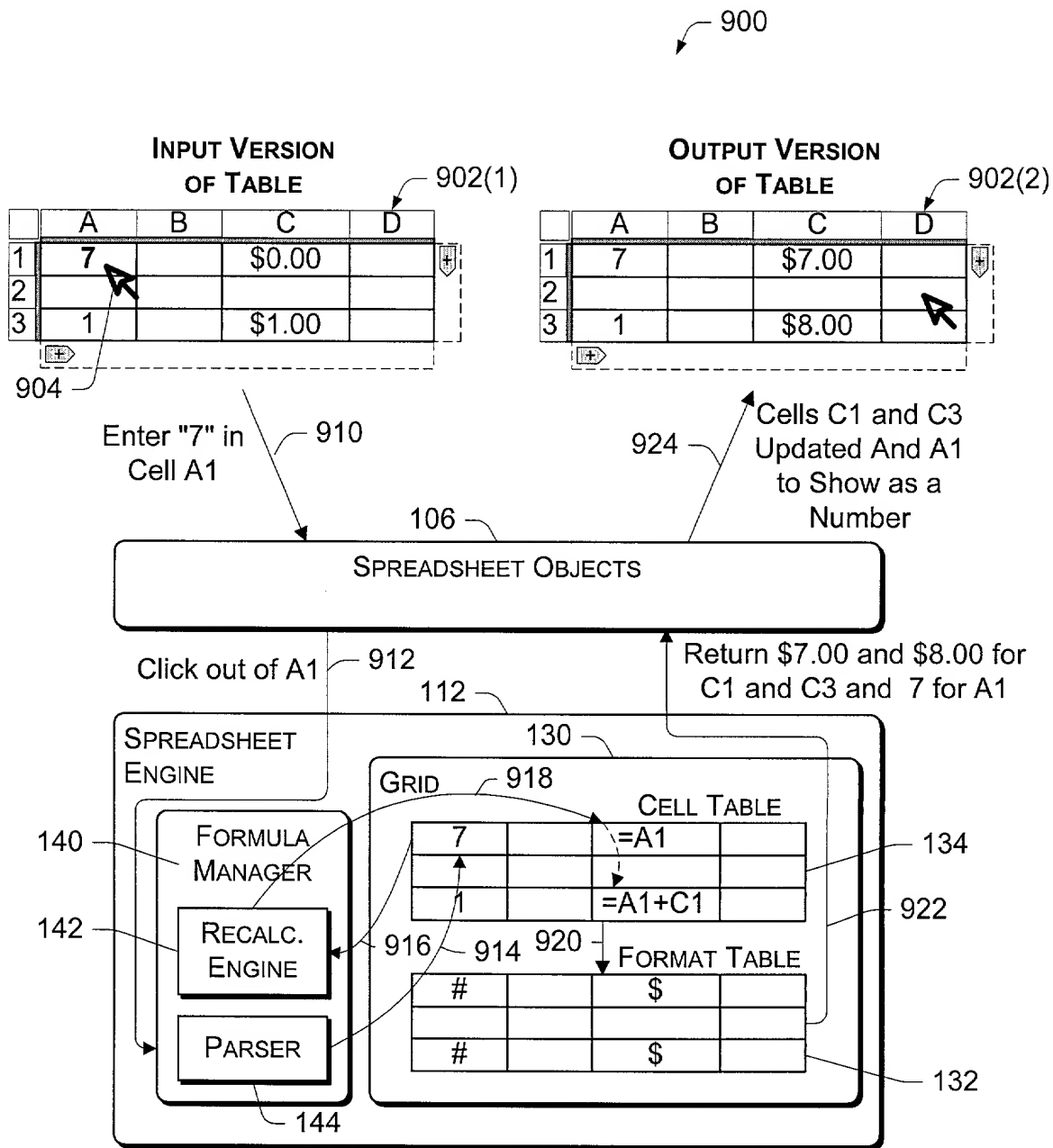


Fig. 9

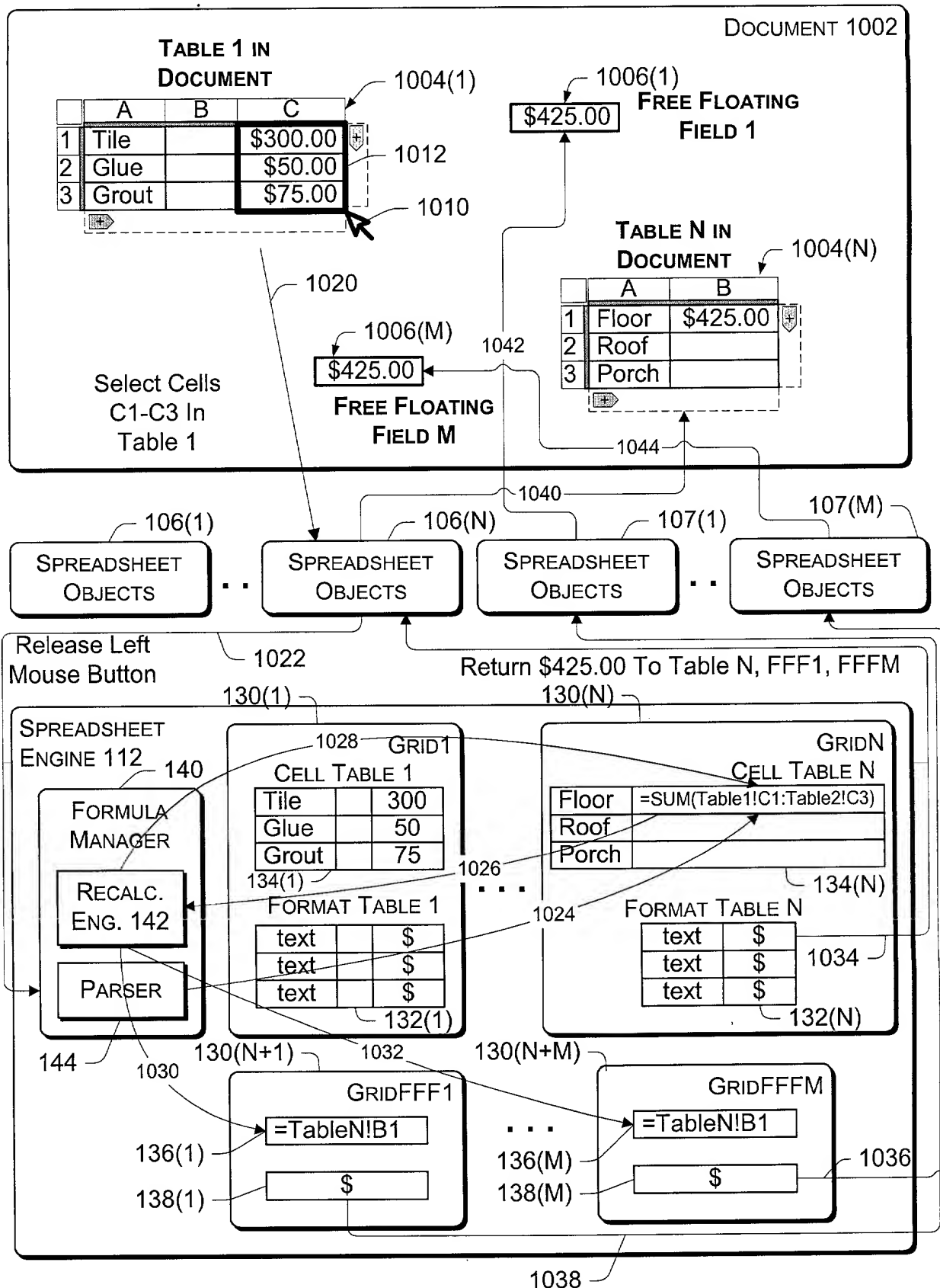


Fig. 10